

## High-Speed Microcontroller Data Book

 **DALLAS**  
SEMICONDUCTOR



© Copyright 1995 by Dallas Semiconductor Corporation. All Rights Reserved. Dallas Semiconductor retains all ownership rights in the technology described herein. Trademarks and registered trademarks of Dallas Semiconductor include each of the following:

Dallas Semiconductor Corporation™	Silicon Label™	1-Wire™	SIP Stik™
Dallas™	Touch Memory™	MicroCan™	Smart Touch Lock™
Dallas Semiconductor™	Touch Thermometer™	Touch Time™	Touch Meter™
DSTM™	Memory Button™	Authorization Button™	Micro Monitor™
Dallastat	Touch Memory Probe™	Touch Pen™	Cyber Card™
Stick'Em Chip™	Certified Dallas Touch™	Time Button™	Cyber Key™
Button Holder™	UniqueWare™	Button Ready PC™	Soft Microcontroller™
Touch Memory EXecutive™	Dallas Registered™	MicroLan™	Secure Microcontroller™
TMEX™	Button™	ID Button™	Soft Silicon™
MultiButton™	Dallas Personal SignOn™	Dallas Protected Software™	All device numbers
TouchMemory Button™	Dallas SignOn™	Load & Lock™	

Dallas Semiconductor has been issued U.S. and foreign patents and has patent applications pending that protect its products, including certain products described in this databook and, in some instances, certain uses, applications, combinations, machines or processes associated with such products. For a complete list of patents issued to Dallas Semiconductor covering the products described herein, please contact the Applications Department at (214) 450-8167. Products may also be protected by other intellectual property rights, including trademark, copyright, mask work and trade secret rights of Dallas Semiconductor.

The Dallas Semiconductor products described in this databook may include copyrighted Dallas Semiconductor computer programs stored in semiconductor memories or other media. Any copyrighted Dallas Semiconductor computer program contained in a Dallas Semiconductor product may not be copied or reproduced in any manner without the express written consent of Dallas Semiconductor. Dallas Semiconductor's sale of the products described in this databook and provision of any supporting or other documentation or technical information or assistance are not intended to convey any license, express or implied, to any copyrights, patents, patent applications or other intellectual property rights of Dallas Semiconductor protecting any combination, machine, process, use or application in which these products might be used. DALLAS SEMICONDUCTOR MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE, EXPRESS OR IMPLIED, REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR THAT THE USE OF ITS PRODUCTS WILL NOT INFRINGE ITS INTELLECTUAL PROPERTY RIGHTS OR THE RIGHTS OF THIRD PARTIES WITH RESPECT TO ANY PARTICULAR USE OR APPLICATION AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY ARISING OUT OF ANY SUCH USE OR APPLICATION, INCLUDING BUT NOT LIMITED TO, CONSEQUENTIAL OR INCIDENTAL DAMAGES.

Dallas Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Dallas Semiconductor product could create a situation where personal injury or death may occur.

Dallas Semiconductor reserves the right to make changes to or discontinue any product or service described herein without notice. Products with data sheets labeled "Preliminary" and other products described herein may not be in production or offered for sale. Dallas Semiconductor advises customers to obtain the current version of the relevant product information before placing orders. Circuit diagrams illustrate typical semiconductor applications and may not be complete. Information published herein supersedes all information regarding this technology published by Dallas Semiconductor in the U.S. before 1995.



# TABLE OF CONTENTS

<b>GENERAL INFORMATION</b> .....	<b>iii</b>
Data Book Cross Reference .....	iv
Sales Offices .....	x
Product Overview .....	xv
Corporate Fact Sheet .....	xix
Quality and Reliability .....	xx
<b>USER'S GUIDE</b> .....	<b>1</b>
Section 1: Introduction .....	2
Section 2: Ordering Information .....	3
Section 3: Architecture .....	4
Section 4: Programming Model .....	7
Section 5: CPU Timing .....	58
Section 6: Memory Access .....	68
Section 7: Power Management .....	79
Section 8: Reset Conditions .....	92
Section 9: Interrupts .....	95
Section 10: Parallel I/O .....	100
Section 11: Programmable Timers .....	104
Section 12: Serial I/O .....	119
Section 13: Timed Access Protection .....	132
Section 14: Real-Time Clock .....	135
Section 15: Battery Backup .....	139
Section 16: Instruction Set Details .....	141
Section 17: Troubleshooting .....	149
<b>DATA SHEETS</b> .....	<b>151</b>
DS80C310 High-Speed Micro .....	152
DS80C320 High-Speed Micro .....	174
DS87C520 EPROM High-Speed Micro .....	206
DS87C530 EPROM Micro with Real Time Clock .....	243
DS80C323 Low-Power Micro .....	281
DS83C520 ROM High-Speed Micro .....	282
<b>APPLICATION NOTES</b> .....	<b>283</b>
Application Note 56 The DS80C320 as a Drop-In Replacement for the 8032 .....	284
Application Note 57 DS80C320 Memory Interface Timing .....	286
Application Note 75 Using the High-Speed Micro's Serial Ports .....	293
Application Note 78 Using Power Management with the DS87C5x0 .....	306
Application Note 79 Using the DS87C530 Real Time Clock .....	333



Application Note 80 Using the High-Speed Micro's Watchdog Timer .....	355
Application Note 81 Memory Expansion with the High-Speed Microcontroller Family .....	361
Application Note 89 High-Speed Micro Memory Interface Timing .....	374
Application Note 91 Microcontroller Design Guidelines for Reducing ALE Signal Noise .....	380
<b>DEVELOPMENT SUPPORT .....</b>	<b>385</b>
<b>INDEX .....</b>	<b>401</b>
<b>DATA SHEETS</b>	
DS80C30 High-Speed Micro .....	123
DS80C30 High-Speed Micro .....	124
DS80C30 EPROM High-Speed Micro .....	125
DS80C30 EPROM Micro with Real Time Clock .....	126
DS80C30 Low-Power Micro .....	127
DS80C30 ROM High-Speed Micro .....	128
<b>APPLICATION NOTES</b>	
Application Note 56 The DS80C30 as a Drop-In Replacement for the 8032 .....	129
Application Note 57 DS80C30 Memory Interface Timing .....	130
Application Note 75 Using the High-Speed Micro's Serial Ports .....	131
Application Note 76 Using Power Management with the DS80C30 .....	132
Application Note 79 Using the DS80C30 Real Time Clock .....	133
<b>USER'S GUIDE</b>	
Section 1: Introduction .....	134
Section 2: Ordering Information .....	135
Section 3: Architecture .....	136
Section 4: Programming Model .....	137
Section 5: CPU Timing .....	138
Section 6: Memory Access .....	139
Section 7: Power Management .....	140
Section 8: Real Conditions .....	141
Section 9: Interrupts .....	142
Section 10: Parallel I/O .....	143
Section 11: Programmable Timers .....	144
Section 12: Serial I/O .....	145
Section 13: Timed Access Protection .....	146
Section 14: Real-Time Clock .....	147
Section 15: Battery Backup .....	148
Section 16: Instruction Set Details .....	149
Section 17: Troubleshooting .....	150
<b>Sales Office .....</b>	
<b>Product Overview .....</b>	
<b>Company Form Sheet .....</b>	
<b>Quality and Reliability .....</b>	



iii



# DATA BOOK CROSS REFERENCE

PART NUMBER	DESCRIPTION	DATA BOOK	PAGE NO.
DS0620	TMEX Professional Developer's Upgrade Kit for DS9092K	Automatic Identification	220
DS1000	5-Tap Silicon Delay Line	System Extension	279
DS1003	4-Tap Silicon Delay Line for RISC Applications	System Extension	285
DS1004	5-Tap High-Speed Silicon Delay Line	System Extension	292
DS1005	5-Tap Silicon Delay Line	System Extension	297
DS1007	7-in-1 Silicon Delay Line	System Extension	302
DS1010	10-Tap Silicon Delay Line	System Extension	307
DS1012	2-in-1 Sub-Miniature Silicon Delay Line with Logic	System Extension	313
DS1013	3-in-1 Silicon Delay Line	System Extension	320
DS1020	Programmable 8-Bit Silicon Delay Line	System Extension	325
DS1021	Programmable 8-Bit Silicon Delay Line	Supplement 4/95	86
DS1033	3-in-1 Low-Voltage Silicon Delay Line	System Extension	334
DS1035	3-in-1 High-Speed Silicon Delay Line	Supplement 4/95	95
DS1040	Programmable One-Shot Pulse Generator	System Extension	346
DS1044	4-in-1 High-Speed Silicon Delay Line	System Extension	352
DS1045	4-Bit Dual Programmable Delay Line	System Extension	358
DS1200	Serial RAM Chip	Timekeeping and NV RAM	834
DS1201	Electronic Tag	Timekeeping and NV RAM	835
DS1202, DS1202S	Serial Timekeeping Chip	Timekeeping and NV RAM	200
DS1204V	Electronic Key	Automatic Identification	266
DS1205S	MultiKey Chip	Automatic Identification	276
DS1205V	MultiKey	Automatic Identification	293
DS1206	Phantom Serial Interface Chip	System Extension	366
DS1207	TimeKey	Automatic Identification	294
DS1210	Nonvolatile Controller Chip	Timekeeping and NV RAM	698
DS1211	Nonvolatile Controller x 8 Chip	Timekeeping and NV RAM	705
DS1212	Nonvolatile Controller x 16 Chip	Timekeeping and NV RAM	706
DS1213B	SmartSocket 16K/64K	Timekeeping and NV RAM	650
DS1213C	SmartSocket 256K	Timekeeping and NV RAM	656
DS1213D	SmartSocket 256K/1M	Timekeeping and NV RAM	658
DS1215	Phantom Time Chip	Timekeeping and NV RAM	211
DS1216B	SmartWatch/RAM 16K/64K	Timekeeping and NV RAM	660
DS1216C	SmartWatch/RAM 64K/256K	Timekeeping and NV RAM	669
DS1216D	SmartWatch/RAM 256K/1M	Timekeeping and NV RAM	670
DS1216E	SmartWatch/ROM 64K/256K	Timekeeping and NV RAM	671
DS1216F	SmartWatch/ROM 64K/256K/1M	Timekeeping and NV RAM	680
DS1217A	Nonvolatile Read/Write Cartridge	Timekeeping and NV RAM	842
DS1217M	Nonvolatile Read/Write Cartridge	Timekeeping and NV RAM	850
DS1218	Nonvolatile Controller	Timekeeping and NV RAM	712
DS1220AB/AD	16K Nonvolatile SRAM	Timekeeping and NV RAM	2
DS1220Y	16K Nonvolatile SRAM	Timekeeping and NV RAM	9
DS1221	Nonvolatile Controller x 4 Chip	Timekeeping and NV RAM	718
DS1222	BankSwitch Chip	System Extension	372
DS1225AB/AD	64K Nonvolatile SRAM	Timekeeping and NV RAM	16
DS1225Y	64K Nonvolatile SRAM	Timekeeping and NV RAM	25
DS1228	+5V Powered Dual RS-232 Transmitter/Receiver	System Extension	262

DS1229	+5V Powered Triple RS-232 Transmitter/Receiver	System Extension	263
DS1230Y/AB	256K Nonvolatile SRAM	Timekeeping and NV RAM	32
DS1231/S	Power Monitor Chip	System Extension	60
DS1232	MicroMonitor Chip	System Extension	69
DS1232LP/LPS	Low Power MicroMonitor Chip	System Extension	76
DS1233	5V EconoReset	System Extension	83
DS1233A	3.3V EconoReset	System Extension	88
DS1234	Conditional Nonvolatile Controller Chip	Timekeeping and NV RAM	726
DS1236	MicroManager Chip	System Extension	98
DS1236A	MicroManager Chip	System Extension	117
DS1237	DRAM Nonvolatizer Chip	Timekeeping and NV RAM	733
DS1238	MicroManager	System Extension	136
DS1238A	MicroManager	System Extension	149
DS1239	MicroManager Chip	System Extension	162
DS1243Y	64K NV SRAM with Phantom Clock	Timekeeping and NV RAM	226
DS1244Y	256K NV SRAM with Phantom Clock	Timekeeping and NV RAM	238
DS1245Y/AB	1024K Nonvolatile SRAM	Timekeeping and NV RAM	41
DS1248Y	1024K NV SRAM with Phantom Clock	Timekeeping and NV RAM	250
DS1249Y	2048K Nonvolatile SRAM	Supplement 4/95	76
DS1250	KeyRing	Timekeeping and NV RAM	858
DS1258K-001	CyberCard Portable Data Carrier Evaluation Kit	Timekeeping and NV RAM	864
DS1258K-002	CyberKey Carrier Evaluation Kit	Timekeeping and NV RAM	865
DS1259	Battery Manager Chip	Timekeeping and NV RAM	743
DS1260	Smart Battery	Timekeeping and NV RAM	748
DS1267	Dual Digital Potentiometer Chip	System Extension	180
DS1275	Line-Powered RS-232 Transceiver Chip	System Extension	268
DS1280	3-Wire to Byte-wide Converter Chip	Timekeeping and NV RAM	920
DS1283	Watchdog Timekeeper Chip	Timekeeping and NV RAM	262
DS1284	Watchdog Timekeeper Chip	Timekeeping and NV RAM	267
DS1285/DS1285Q	Real Time Clock	Timekeeping and NV RAM	270
DS1286	Watchdog Timekeeper	Timekeeping and NV RAM	271
DS1287	Real Time Clock	Timekeeping and NV RAM	283
DS1287A	Real Time Clock	Timekeeping and NV RAM	284
DS12885, DS12885Q, and DS12885T	Real Time Clock	Timekeeping and NV RAM	285
DS12887	Real Time Clock	Timekeeping and NV RAM	291
DS12887A	Real Time Clock	Timekeeping and NV RAM	308
DS129x	Eliminator	System Extension	376
DS1302	Trickle Charge Timekeeping Chip	Timekeeping and NV RAM	310
DS1307	64 X 8 Serial Real Time Clock	Supplement 4/95	208
DS1330YLPM/ABLPM	256K Nonvolatile SRAM with Power Monitors	Timekeeping and NV RAM	49
DS1336	Afterburner Chip	System Extension	382
DS1345YLPM/ABLPM	1024K Nonvolatile SRAM with Power Monitors	Timekeeping and NV RAM	50
DS1350YLPM/ABLPM	4096K Nonvolatile SRAM with Power Monitors	Timekeeping and NV RAM	51
DS1380	RAMport	Timekeeping and NV RAM	52
DS1381	NV RAMport	Timekeeping and NV RAM	55



# DATA BOOK CROSS REFERENCE

DS1385/DS1387	RAMified Real Time Clock 4K x 8	Timekeeping and NV RAM	322
DS1386	RAMified Watchdog Timekeeper	Timekeeping and NV RAM	342
DS1395/DS1397	RAMified Real Time Clock	Timekeeping and NV RAM	355
DS1401	Front Panel Button Holder	Automatic Identification	126, 312
DS1402	Button Cable	Automatic Identification	126, 312
DS1410D	Parallel Port Button Holder	Automatic Identification	313
DS1412	Serial Port Button Holder	Automatic Identification	315
DS1414	Network Button Holder	Automatic Identification	320
DS1420	Serial ID Button	Automatic Identification	321
DS1422	1Kbit Add-Only UniqueWare™ Button	Automatic Identification	322
DS1425	Multi Button™	Automatic Identification	324
DS1427	Time Button™	Automatic Identification	326
DS1410K	Parallel Holder Developer's Kit	Automatic Identification	328
DS1412K	Serial Holder Developer's Kit	Automatic Identification	329
DS1414K	Network Holder Developer's Kit	Automatic Identification	330
DS14285/DS14287	Real Time Clock with NVRAM Control	Supplement 4/95	219
DS1485/DS1488	RAMified Real Time Clock 8K x 8	Timekeeping and NV RAM	374
DS1486	RAMified Watchdog Timekeeper	Timekeeping and NV RAM	394
DS1495/DS1497	RAMified Real Time Clock	Timekeeping and NV RAM	407
DS1585/DS1587/DS1587LPM	Serialized Real Time Clocks	Timekeeping and NV RAM	426
DS1589/DS1593	Serialized Real Time Clocks	Timekeeping and NV RAM	455
DS1602	Elapsed Time Counter	Timekeeping and NV RAM	468
DS1603	Elapsed Time Counter Module	Timekeeping and NV RAM	477
DS1608	EconoRAM Time Chip	Timekeeping and NV RAM	485
DS1609	Dual Port RAM	Timekeeping and NV RAM	931
DS1610	Partitioned NV Controller	Timekeeping and NV RAM	756
DS1611	Programmable System Monitor	System Extension	387
DS1612	Lithium Battery Monitor	Timekeeping and NV RAM	766
DS1613C	Partitioned SmartSocket 256K	Timekeeping and NV RAM	681
DS1613D	Partitioned SmartSocket 1M	Timekeeping and NV RAM	689
DS1620	Digital Thermometer and Thermostat	Supplement 4/95	130
DS1621	Digital Thermometer and Thermostat	Supplement 4/95	140
DS1625	Digital Thermometer and Thermostat	Supplement 4/95	153
DS1630Y/AB,			
DS1630Y/ABLPM	Partitionable 256K NV SRAM	Timekeeping and NV RAM	63
DS1632	PC Power Fail and Reset Controller	System Extension	166
DS1633	High-Speed Battery Recharger	System Extension	2
DS1633x	High-Speed Battery Charger	Supplement 4/95	2
DS1640/DS1640C	Personal Computer Power FET	System Extension	400
DS1642	Nonvolatile Timekeeping RAM	Timekeeping and NV RAM	511
DS1643/DS1643LPM	Nonvolatile Timekeeping RAM	Timekeeping and NV RAM	521
DS1644/DS1644LPM	Nonvolatile Timekeeping RAM	Timekeeping and NV RAM	533
DS1645Y/AB,			
DS1645Y/ABLPM	Partitionable 1024K NV SRAM	Timekeeping and NV RAM	75
DS1646/DS1646LPM	Nonvolatile Timekeeping RAM	Timekeeping and NV RAM	544
DS1647	Nonvolatile Timekeeping RAM	Timekeeping and NV RAM	555
DS1650Y/AB,			
DS1650Y/ABLPM	Partitionable 4096K NV SRAM	Timekeeping and NV RAM	87
DS1651/DS1652	3-Code Lock/Key Match Memory System	System Extension	404
DS1652B	Code Memory Key	System Extension	414

DS1653/DS1652	4-Code Lock/Key Match Memory System	System Extension	421
DS1658Y/AB	Partitionable 128K x 16 NV SRAM	Timekeeping and NV RAM	99
DS1666, DS1666S	Audio Digital Resistor	System Extension	191
DS1667	Digital Resistor with OP AMP	System Extension	196
DS1668, DS1669, DS1669S	Dallastat™ Electronic Digital Rheostat	System Extension	207
DS1685/DS1687	3 Volt/5 Volt Real Time Clock	Supplement 4/95	242
DS1688/DS1691	3 Volt/5 Volt Serialized Real Time Clock with NVRAM Control	Supplement 4/95	274
DS1689/DS1693	5 Volt/3 Volt Serialized Real Time Clock with NV RAM Control	Supplement 4/95	279
DS1710	Partitioned NV Controller	Timekeeping and NV RAM	779
DS17285/DS17287	3 Volt/5 Volt Real Time Clock	Supplement 4/95	309
DS1730Y/YLPM	3 Volt Partitionable 256K NV SRAM	Timekeeping and NV RAM	110
DS1745Y/YLPM	3 Volt Partitionable 1024K NV SRAM	Timekeeping and NV RAM	122
DS17485/DS17487	3 Volt/5 Volt Real Time Clock	Supplement 4/95	340
DS1750Y/YLPM	3 Volt Partitionable 4096K NV SRAM	Timekeeping and NV RAM	134
DS1758Y	3V Partitionable 128K x 16 NV SRAM	Timekeeping and NV RAM	146
DS1801	Dual Audio Taper Potentiometer	Supplement 4/95	16
DS1802	Dual Audio Taper Potentiometer with Pushbutton Control	System Extension	217
DS1803	Addressable Dual Digital Potentiometer	Supplement 4/95	25
DS1820	1-Wire™ Digital Thermometer	Supplement 4/95	166
DS1821	Programmable Digital Thermostat	Supplement 4/95	193
DS1830	Programmable MicroMonitor	System Extension	173
DS1832	3.3 Volt MicroMonitor Chip	Supplement 4/95	8
DS1833	5V EconoReset	System Extension	174
DS1837	Quick Battery Recharger	System Extension	13
DS1867	Dual Digital Potentiometer with EEPROM	System Extension	232
DS1868	Dual Digital Potentiometer Chip	System Extension	243
DS1869	3V Dallastat™ Electronic Digital Rheostat	System Extension	254
DS1920	Touch Thermometer™	Automatic Identification	107
DS1990A	Touch Serial Number™	Automatic Identification	2
DS1991	Touch MultiKey™	Automatic Identification	12
DS1992/DS1993	1Kbit/4Kbit Touch Memory™	Automatic Identification	26
DS1994	4Kbit Plus Time Touch Memory™	Automatic Identification	26
DS1995	16Kbit Touch Memory™	Automatic Identification	46
DS1996	64Kbit Touch Memory™	Automatic Identification	47
DS1982	1Kbit Add-Only Touch Memory™	Automatic Identification	62
DS1985	16Kbit Add-Only Touch Memory™	Automatic Identification	83
DS1986	64Kbit Add-Only Touch Memory™	Automatic Identification	106
DS2009	512 x 9 FIFO Chip	Timekeeping and NV RAM	938
DS2010	1024 x 9 FIFO Chip	Timekeeping and NV RAM	952
DS2011	2048 x 9 FIFO Chip	Timekeeping and NV RAM	953
DS2012	4096 x 9 FIFO Chip	Timekeeping and NV RAM	954
DS2013	8192 x 9 FIFO Chip	Timekeeping and NV RAM	955
DS2016	2K x 8 3V Operation Static RAM	Timekeeping and NV RAM	794
DS2064	8K x 8 3V Operation Static RAM	Timekeeping and NV RAM	804
DS21S07A	SCSI Terminator	Supplement 4/95	102
DS2108	Differential SCSI Switchable Terminator	Supplement 4/95	109
DS2109	Plug and Play SCSI Terminator	Supplement 4/95	114
DS2112	BTL Terminator	Supplement 4/95	124

# DATA BOOK CROSS REFERENCE

DS2130Q	Voice Messaging Processor	Telecommunications	8
DS2132A/Q	Digital Answering Machine Processor	Telecommunications	30
DS2141A	T1 Controller	Telecommunications	246
DS21Q41AFP	Quad T1 Controller	Telecommunications	281
DS2143/DS2143Q	E1 Controller	Telecommunications	283
DS21Q43FP	Quad E1 Controller	Telecommunications	323
DS2151Q	T1 Single-Chip Transceiver	Telecommunications	80
DS2153Q	E1 Single-Chip Transceiver	Telecommunications	126
DS2165/DS2165Q	16/24/32Kbps ADPCM Processor	Telecommunications	47
DS2167/DS2168	ADPCM Processor	Telecommunications	64
DS2180A	T1 Transceiver	Telecommunications	325
DS2181A	CEPT Primary Rate Transceiver	Telecommunications	361
DS2182	T1 Line Monitor	Telecommunications	393
DS2186	Transmit Line Interface	Telecommunications	176
DS2187	Receive Line Interface	Telecommunications	187
DS2188	T1/CEPT Jitter Attenuator	Telecommunications	196
DS2190-003	T1 Network Interface Unit (NIU)	Telecommunications	208
DS22B57	32K x 8 Static RAM	Timekeeping and NV RAM	814
DS222	Dual RS-232 Transmitter/Receiver with Shutdown	Supplement 4/95	36
DS2223/DS2224	EconoRAM	Automatic Identification	209
DS2227	Flexible NV SRAM Stik	Timekeeping and NV RAM	157
DS2229	Word-Wide 8 Meg SRAM Stik	Timekeeping and NV RAM	823
DS2250(T)	Soft Microcontroller	Soft Microcontroller	176
DS2251(T)	128K Soft Microcontroller	Soft Microcontroller	195
DS2252(T)	Secure Microcontroller	Soft Microcontroller	215
DS229	RS-232 Transmitter/Receiver	Supplement 4/95	48
DS2290	T1 Isolation Stik	Telecommunications	225
DS2291	T1 Long Loop Stik	Telecommunications	235
DS232A	Dual RS-232 Transmitter/Receiver	Supplement 4/95	57
DS233A	Dual RS-232 Transmitter/Receiver	Supplement 4/95	66
DS2401	Silicon Serial Number	Automatic Identification	140
DS2404	EconoRAM Time Chip	Timekeeping and NV RAM	594
DS2404S-C01	Dual Port Memory Plus Time	Automatic Identification	206
DS2405	Addressable Switch	Automatic Identification	194
DS2434	Battery Identification Chip	System Extension	21
DS2435	Battery Identification Chip with Time/Temperature Histogram	System Extension	38
DS2502	1Kbit Add-Only Memory	Automatic Identification	149
DS2505	16Kbit Add-Only Memory	Automatic Identification	170
DS2506	64Kbit Add-Only Memory	Automatic Identification	193
DS5000(T)	Soft Microcontroller	Soft Microcontroller	229
DS5000FP	Soft Microcontroller Chip	Soft Microcontroller	247
DS5000TK	Evaluation Kit	Soft Microcontroller	325
DS5001FP	128K Soft Micro Chip	Soft Microcontroller	267
DS5002FP	Secure Micro	Soft Microcontroller	290
DS620x	CyberKey	Timekeeping and NV RAM	866
DS6417	CyberCard EV 4M-Bit NV SRAM	Timekeeping and NV RAM	893
DS9000	Bytewise Cable Harness	Timekeeping and NV RAM	903
DS9002	Cartridge Housing	Timekeeping and NV RAM	904
DS9003	Cartridge Proto Board	Timekeeping and NV RAM	905



DS908xx	CyberKey/Card Receptacles	Timekeeping and NV RAM	906
DS9092	Touch Memory Probe	Automatic Identification	127
DS9092K	Touch Memory Starter Kit	Automatic Identification	221
DS9092R	Touch Port	Automatic Identification	129
DS9093x	Touch Memory Mount Products	Automatic Identification	130
DS9094	MicroCan Clip	Automatic Identification	132
DS9096P	Touch Memory Adhesive Pads	Automatic Identification	133
DS9097/DS9097E	Touch COM Port Adapter	Automatic Identification	134
DS9098	MicroCan Retainer	Automatic Identification	135
DS9099K	Touch Pen Development Kit	Automatic Identification	222
DS9099	Touch Pen Chip Set	Automatic Identification	222
DS9100	Touch and Hold Probe Stampings	Automatic Identification	136
DS9101	Multi-Purpose Clip	Automatic Identification	137
DS9103K	Touch Memory Access Control Demo Kit	Automatic Identification	224

Idaho	Canada	HEADQUARTERS
Western Tech Sales	Devotee Marketing	Dallas, TX
Boise, ID	Bumby - British Columbia	Dallas, TX
(208) 376-8700	(604) 430-5580	Voice (214) 450-0448
Illinois	Calgary, Alberta	FAX (214) 450-0470
Sumit Inc.	(403) 250-3084	Great Lakes Area
Rolling Meadows, IL	Dynamic Components	Carmel, IN
(708) 881-8500	Ottawa, Ontario	(317) 573-7873
Qip Technology Sales	(613) 580-8800	Mid Atlantic
St. Louis, MO	Pointe Claire, Quebec	Methuen, NJ
(314) 880-0443	(514) 684-7353	(603) 586-1219
Indiana	Colorado	North Central
Valentine Associates, Inc.	Whipman Assoc.	Hollman Estates, IL
Carmel, IN	West Ridge, CO	(708) 480-6378
(317) 845-0000	(303) 423-1020	Northeast E. Canada
Kansas	Connecticut	Chelmsford, MA
Qip Technology Sales	Technology Sales Inc.	(603) 280-1055
Kansas City, MO	Wilmington, CT	South Central/West
(816) 483-8785	(203) 288-8883	Dallas, TX
Kentucky	Delaware	(214) 788-2197
Glen White & Assoc.	S. J. Mid Atlantic	Southeast
Huntsville, AL	Mr. Laurel, NJ	Duluth, GA
(205) 882-8787	(800) 888-1234	(404) 623-5513
Louisiana	Florida	Southwest
West Assoc.	Wilton Inc.	Irvine, CA
Richardson, TX	Davis, FL	(714) 724-4223
(214) 880-5800	(800) 370-6998	Western Area
Maine	Indianapolis, IN	Sunnyvale, CA
Wilt-Gorm Assoc.	(407) 777-3388	(408) 733-8400
Woods, MA	Tampa, FL	South Florida
(617) 882-3217	(813) 287-1433	St. Petersburg, FL
Maryland	Georgia	(813) 527-6454
S. J. Chesapeake	Glen White & Assoc.	NEEDHAM HILLS
Pots Church, VA	Duluth, GA	Boston, England
(703) 882-2233	(404) 418-1800	44-1-21-780-5828
		(616) 885-6222

## SALES OFFICES

### I'm Interested. Who Do I Call?

For technical product information, call (214) 450-0448 or FAX us at (214) 450-3715.  
If calling from overseas, dial (214) 450-5351. We can mail a data book and product literature immediately, or we can have you talk to an applications engineer.

You can order any Dallas Semiconductor product for next-day shipment with a Visa, MasterCard, or American Express. Call our Credit Card Sales service at 1-800-336-6933.

You can also contact our nearest distributor, representative, or sales office.

Literature & Technical Support ..... 1-214-450-0448  
Sales & Customer Service ..... 1-214-450-0969  
Automatic Data Sheet Faxback ..... 1-214-450-0441  
Corporate FTP Site ..... ftp.dalsemi.com  
Worldwide Web Site ..... <http://www.dalsemi.com>  
E-Mail Format ..... [firstname.lastname@dalsemi.com](mailto:firstname.lastname@dalsemi.com)  
Sales ..... sales@dalsemi.com

#### HEADQUARTERS

Dallas, TX  
Voice: (214) 450-0448  
FAX: (214) 450-0470

#### Great Lakes Area

Carmel, IN  
(317) 573-7613

#### Mid Atlantic

Marlton, NJ  
(609) 596-1919

#### North Central

Hoffman Estates, IL  
(708) 490-5378

#### Northeast/E. Canada

Chelmsford, MA  
(508) 256-4995

#### South Central/West

Dallas, TX  
(214) 788-2197

#### Southeast

Duluth, GA  
(404) 623-5813

#### Southwest

Irvine, CA  
(714) 724-4523

#### Western Area

Sunnyvale, CA  
(408) 733-8400

#### South/Florida

St. Petersburg, FL  
(813) 527-6454

#### EUROPEAN OFFICE

Birmingham, England  
44-1-21-782-2959

#### PACIFIC RIM OFFICE

Dallas, TX  
(214) 450-5363

#### SOUTH AMERICAN OFFICE

Dallas, TX  
(214) 788-2197

#### NORTH AMERICAN SALES REPRESENTATIVES

##### Alabama

*Glen White & Assoc.*  
Huntsville, AL  
(205) 882-6751

##### Arizona

*System Sales of AZ*  
Mesa, AZ  
(602) 464-9989

##### Arkansas

*West Associates*  
Tulsa, OK  
(918) 492-4300

##### California

*Harvey King, Inc.*  
San Diego, CA  
(619) 695-9300  
*R.S.V.P. Associates*  
Sunnyvale, CA  
(408) 467-1200

##### S C Cubed

Tustin, CA  
(714) 731-9206  
Westlake Village, CA  
(818) 865-6222

#### Canada

*Davetek Marketing*  
Burnaby, British Columbia  
(604) 430-3680

Calgary, Alberta

(403) 250-2034

#### *Dynasty Components*

Ottawa, Ontario  
(613) 596-9800  
Pointe Claire, Quebec  
(514) 694-7353

#### Colorado

*Waugaman Assoc.*  
Wheat Ridge, CO  
(303) 423-1020

#### Connecticut

*Technology Sales Inc.*  
Wallingford, CT  
(203) 269-8853

#### Delaware

*S-J Mid-Atlantic*  
Mt. Laurel, NJ  
(609) 866-1234

#### Florida

*Naltron Inc.*  
Davie, FL  
(305) 370-9396  
Indialantic, FL  
(407) 777-3399  
Tampa, FL  
(813) 287-1433

#### Georgia

*Glen White & Assoc.*  
Duluth, GA  
(404) 418-1500

#### Idaho

*Western Tech. Sales*  
Boise, ID  
(208) 376-8700

#### Illinois

*Sumer Inc.*  
Rolling Meadows, IL  
(708) 991-8500  
*Gibb Technology Sales*  
St. Louis, MO  
(314) 890-0443

#### Indiana

*Valentine Associates, Inc.*  
Carmel, IN  
(317) 846-0008

#### Kansas

*Gibb Technology Sales*  
Kansas City, MO  
(816) 483-6785

#### Kentucky

*Glen White & Assoc.*  
Huntsville, AL  
(205) 882-6751

#### Louisiana

*West Assoc.*  
Richardson, TX  
(214) 680-2800

#### Maine

*Mill-Bern Assoc.*  
Woburn, MA  
(617) 932-3311

#### Maryland

*S-J Chesapeake*  
Falls Church, VA  
(703) 533-2233

**Massachusetts****Mill-Bern Assoc.**

Woburn, MA  
(617) 932-3311

**Michigan****Trilogy Marketing**

Auburn Hills, MI  
(810) 377-4900

**Minnesota****Cahill, Schmitz & Cahill**

St. Paul, MN  
(612) 646-7217

**Mississippi****Glen White & Assoc.**

Huntsville, AL  
(205) 882-6751

**Missouri****Gibb Technology Sales**

St. Louis, MO  
(314) 890-0443

**Montana****Waugaman Assoc.**

Wheat Ridge, CO  
(303) 423-1020

**Nebraska****Waugaman Associates**

Wheat Ridge, CO  
(303) 423-1020

**Nevada**

Bay Area/Northern CA,NV

Sunnyvale, CA  
(408) 733-8400

**System Sales of AZ**

Mesa, AZ  
(602) 464-9989

**New Hampshire****Mill-Bern Assoc.**

Woburn, MA  
(617) 932-3111

**New Jersey****S-J Associates**

Mt. Laurel, NJ  
(609) 866-1234  
Rockville Centre, NY  
(516) 536-4242

**New Mexico****System Sales of AZ**

Albuquerque, NM  
(505) 889-2901

**New York****S-J Associates**

Rockville Centre, NY  
(516) 536-4242

**Technology Sales Inc.**

Fairport, NY  
(716) 223-7500

**North Carolina****Glen White & Assoc.**

Huntersville, NC  
(704) 875-3777  
Raleigh, NC  
(919) 848-1931

**North Dakota****Cahill, Schmitz & Cahill**

St. Paul, MN  
(612) 646-7217

**Ohio****Millennium Tech. Sales**

Cincinnati, OH  
(513) 871-2424  
Dublin, OH  
(614) 793-9545  
Mayfield Village, OH  
(216) 461-3500

**Oklahoma****West Associates**

Tulsa, OK  
(918) 492-4300

**Oregon****Western Tech. Sales**

Beaverton, OR  
(503) 644-8860

**Pennsylvania****Millennium Tech. Sales**

Cincinnati, OH  
(513) 871-2424  
**S-J Associates**  
Mt. Laurel, NJ  
(609) 866-1234

**Puerto Rico**

Tampa, FL  
(813) 287-1433

**Rhode Island****Mill-Bern Assoc.**

Woburn, MA  
(617) 932-3311

**South Carolina****Glen White & Assoc.**

Huntersville, NC  
(704) 875-3777

**South Dakota****Cahill, Schmitz & Cahill**

St. Paul, MN  
(612) 646-7217

**Tennessee****Glen White & Assoc.**

Huntsville, AL  
(205) 882-6751

**Texas****West Associates**

Austin, TX  
(512) 343-1199  
Houston, TX  
(713) 999-0101  
Richardson, TX  
(214) 680-2800

**Utah****Waugaman Associates**

Salt Lake City, UT  
(801) 261-0802

**Vermont****Mill-Bern Assoc.**

Woburn, MA  
(617) 932-3311

**Virginia****S-J Chesapeake**

Falls Church, VA  
(703) 533-2233

**Washington****Western Tech. Sales**

Bellevue, WA  
(206) 641-3900  
Spokane, WA  
(509) 922-7600

**West Virginia****S-J Chesapeake**

Falls Church, VA  
(703) 533-2233

**Wisconsin****Cahill, Schmitz & Cahill**

St. Paul, MN  
(612) 646-7217  
**Sumer, Inc.**  
Brookfield, WI  
(414) 784-6641

**Wyoming****Waugaman Assoc.**

Wheat Ridge, CO  
(303) 423-1020

**NORTH AMERICAN DISTRIBUTORS****■ ADDED VALUE**

**Electronic Dist., Inc.**  
(AVED)

**Arizona**

Scottsdale, AZ  
(602) 951-9788

**California**

San Diego, CA  
(619) 558-8890  
Tustin, CA  
(714) 573-5000  
Visalia, CA  
(209) 734-8861

**Colorado**

Wheat Ridge, CO  
(303) 422-1701

**Utah**

Midvale, UT  
(801) 975-9500

**■ ADVENT ELECTRONICS****Illinois**

Des Plaines, IL  
(708) 297-6200

**Indiana**

Indianapolis, IN  
(317) 872-4910

**Iowa**

Cedar Rapids, IA  
(319) 363-0221

**Michigan**

Farmington Hills, MI  
(313) 477-1650

**■ ALMAC ELECTRONICS****Oregon**

Beaverton, OR  
(503) 690-5613

**Washington**

Bellevue, WA  
(206) 643-9992

**■ ANTHEM ELECTRONICS****Alabama**

Huntsville, AL  
(205) 890-0302

**Arizona**

Tempe, AZ  
(602) 966-6600

**California**

Chatsworth, CA  
(818) 775-0410



Irvine, CA  
(714) 768-4444  
Rocklin, CA  
(916) 624-9744  
San Diego, CA  
(619) 453-9005  
San Jose, CA  
(408) 453-1200

**Colorado**  
Englewood, CO  
(303) 790-4500

**Connecticut**  
Waterbury, CT  
(203) 575-1575

**Georgia**  
Duluth, GA  
(404) 931-3900

**Illinois**  
Schaumburg, IL  
(708) 884-0200

**Maryland**  
Columbia, MD  
(410) 995-6640

**Massachusetts**  
Wilmington, MA  
(508) 657-5170

**Minnesota**  
Eden Prairie, MN  
(612) 944-5454

**New Jersey**  
Pine Brook, NJ  
(201) 227-7960

**New York**  
Commack, NY  
(516) 864-6600

**Oregon**  
Beaverton, OR  
(503) 643-1114

**Pennsylvania**  
Horsham, PA  
(215) 443-5150

**Texas**  
Austin, TX  
(512) 388-0049  
Richardson, TX  
(214) 238-7100

**Utah**  
Salt Lake City, UT  
(801) 973-8555

**Washington**  
Bothell, WA  
(206) 483-1700

# **■ FARNELL**

## **Canada**

Burnaby, British Columbia  
(604) 421-6222  
Calgary, Alberta  
(403) 273-2780  
Concord, Ontario  
(416) 798-4884  
Nepean, Ontario  
(613) 596-6980  
Pointe Claire, Quebec  
(514) 697-8149  
Winnipeg, Manitoba  
(403) 273-2780

# **■ FUTURE ELECTRONICS**

## **Alabama**

Huntsville, AL  
(205) 830-2322

## **Arizona**

Phoenix, AZ  
(602) 968-7140

## **California**

Agoura Hills, CA  
(818) 865-0040  
Irvine, CA  
(714) 453-1515  
San Diego, CA  
(619) 625-2800  
San Jose, CA  
(408) 434-1122

## **Canada**

Calgary, Alberta  
(403) 250-5550  
Edmonton, Alberta  
(403) 438-2858  
Mississauga, Ontario  
(905) 612-9200  
Ottawa, Ontario  
(613) 820-8313  
Pointe Claire, Quebec  
(514) 694-7710  
Quebec City, Quebec  
(418) 877-6666  
Winnipeg, Manitoba  
(204) 944-1446

## **Colorado**

Lake Wood, CO  
(303) 232-2008

## **Connecticut**

Cheshire, CT  
(203) 250-0083

## **Florida**

Altamonte Springs, FL  
(407) 865-7900

Deerfield Beach, FL  
(305) 426-4043  
Largo, FL  
(813) 530-1222

## **Georgia**

Norcross, GA  
(404) 441-7676

## **Illinois**

Hoffman Estates, IL  
(708) 882-1255

## **Indiana**

Indianapolis, IN  
(317) 469-0447

## **Kansas**

Overland Park, KS  
(913) 649-1531

## **Maryland**

Columbia, MD  
(410) 290-0600

## **Massachusetts**

Bolton, MA  
(508) 779-3000

## **Michigan**

Livonia, MI  
(313) 261-5270  
Grand Rapids, MI  
(616) 698-6800

## **Minnesota**

Eden Prairie, MN  
(612) 944-2200

## **Missouri**

St. Louis, MO  
(314) 469-6805

## **New Jersey**

Marlton, NJ  
(609) 596-4080  
Parsippany, NJ  
(201) 299-0400

## **New York**

Hauppauge, NY  
(516) 234-4000  
N. Syracuse, NY  
(315) 451-2371  
Rochester, NY  
(716) 387-9550

## **North Carolina**

Charlotte, NC  
(704) 547-1107  
Raleigh, NC  
(919) 790-7111

## **Ohio**

Beavercreek, OH  
(513) 426-0090

Mayfield Heights, OH  
(216) 449-6996

## **Oregon**

Beaverton, OR  
(503) 645-6454

## **Texas**

Austin, TX  
(512) 502-0991  
Houston, TX  
(713) 785-1155  
Richardson, TX  
(214) 437-2437

## **Utah**

Salt Lake City, UT  
(801) 467-4448

## **Washington**

Bothell, WA  
(206) 489-3400

## **Wisconsin**

Brookfield, WI  
(414) 879-0244

# **■ HAMILTON-HALLMARK**

## **Alabama**

Huntsville, AL  
(205) 837-8700

## **Arizona**

Phoenix, AZ  
(602) 437-1200

## **California**

Costa Mesa, CA  
(714) 641-4100  
Rocklin, CA  
(916) 624-9781  
San Diego, CA  
(619) 571-7540  
San Jose, CA  
(408) 435-3500  
Woodland Hills, CA  
(818) 594-0404

## **Colorado**

Colorado Springs, CO  
(719) 637-0055  
Englewood, CO  
(303) 790-1662

## **Connecticut**

Cheshire, CT  
(203) 271-2844

## **Florida**

Ft. Lauderdale, FL  
(305) 484-5482  
Largo, FL  
(813) 541-7440

Winter Park, FL  
(407) 657-3300

### Georgia

Duluth, GA  
(404) 623-4400

### Illinois

Bensonville, IL  
(708) 860-7780

### Indiana

Carmel, IN  
(317) 575-3500

### Kansas

Lenexa, KS  
(913) 888-4747

### Kentucky

Lexington, KY  
(800) 235-6039

### Maryland

Columbia, MD  
(410) 988-9800

### Massachusetts

Peabody, MA  
(508) 532-9808

### Michigan

Novi, MI  
(313) 347-4271  
Plymouth, MI  
(313) 416-5800

### Minnesota

Bloomington, MN  
(612) 881-2600

### Missouri

Earth City, MO  
(314) 291-5350

### New Jersey

Cherry Hill, NJ  
(609) 424-0110  
Parsippany, NJ  
(201) 515-3000

### New York

Hauppauge, NY  
(516) 434-7474  
Hauppauge, NY  
(516) 434-7400  
Rochester, NY  
(716) 475-9130

### North Carolina

Raleigh, NC  
(919) 872-0712

### Ohio

Dayton, OH  
(513) 439-6735  
Solon, OH  
(216) 498-1100

Worthington, OH  
(614) 888-3313

### Oklahoma

Tulsa, OK  
(918) 254-6110

### Oregon

Beaverton, OR  
(503) 526-6200

### Texas

Austin, TX  
(512) 258-8848  
Dallas, TX  
(214) 553-4300  
Houston, TX  
(713) 781-6100

### Utah

Salt Lake City, UT  
(801) 266-2022

### Washington

Redmond, WA  
(206) 881-6697

### Wisconsin

New Berlin, WI  
(414) 797-7844

## ■ INSIGHT ELECTRONICS

### Alabama

Huntsville, AL  
(205) 830-1222

### Arizona

Tempe, AZ  
(602) 829-1800

### California

Irvine, CA  
(714) 727-3291  
San Diego, CA  
(619) 677-3100  
Sunnyvale, CA  
(408) 720-9222  
Westlake Village, CA  
(818) 707-2101

### Colorado

Englewood, CO  
(303) 649-1800

### Georgia

Duluth, GA  
(404) 717-8566

### Illinois

Schaumburg, IL  
(708) 885-9700

### Massachusetts

Burlington, MA  
(617) 270-9400

### Minnesota

St. Louis Park, MN  
(612) 525-9999

### Ohio

Valley View, OH  
(216) 520-4333

### Oregon

Beaverton, OR  
(503) 644-3300

### Texas

Austin, TX  
(512) 719-3090  
Richardson, TX  
(214) 783-0800

### Washington

Kirkland, WA  
(206) 820-8100

### Wisconsin

Wauwatosa, WI  
(414) 258-5338

## ■ INTERFACE ELECTRONICS

### Massachusetts

Hopkinton, MA  
(508) 435-0100

## ■ MILGRAY ELECTRONICS

### Alabama

Huntsville, AL  
(205) 722-9709

### California

Irvine, CA  
(714) 753-1282  
San Diego, CA  
(619) 457-7545  
San Jose, CA  
(408) 456-0900  
Thousand Oaks, CA  
(805) 371-9399

### Canada

Mississauga, ON  
(905) 678-0958  
Pointe Claire, Quebec  
(514) 426-5900

### Colorado

Englewood, CO  
(303) 721-7702

### Connecticut

Milford, CT  
(203) 878-5538

### Florida

Lake Mary, FL  
(407) 321-2555

### Georgia

Norcross, GA  
(404) 446-9777

### Illinois

Palatine, IL  
(708) 202-1900

### Indiana

Indianapolis, IN  
(317) 781-9997

### Kansas

Overland Park, KS  
(913) 236-8800

### Maryland

Columbus, MD  
(410) 730-6119

### Massachusetts

Wilmington, MA  
(508) 657-5900

### New Jersey

Marlton, NJ  
(609) 983-5010  
Parsippany, NJ  
(201) 335-1766

### New York

Farmingdale, NY  
(516) 391-3000  
Pittsford, NY  
(716) 381-9700

### North Carolina

Raleigh, NC  
(919) 790-8094

### Ohio

Cleveland, OH  
(216) 447-1520

### Texas

Austin, TX  
(512) 331-9961  
Dallas, TX  
(214) 248-1603  
Stafford, TX  
(713) 240-5360

### Utah

Murray, UT  
(801) 261-2999

## ■ NEWARK ELECTRONICS

### Illinois

Chicago, IL  
(312) 907-5436

## ■ REPTRON ELECTRONICS

### Florida

Tampa, FL  
(813) 854-2351

■ **STERLING ELECTRONICS****Arizona**

Phoenix, AZ  
(602) 437-5565

**California**

Irvine, CA  
(714) 453-7660  
San Diego, CA  
(619) 560-8097  
San Jose, CA  
(408) 435-0835  
Westlake Village, CA  
(818) 865-2333

**Colorado**

Englewood, CO  
(303) 792-3939

**Connecticut**

Wallingford, CT  
(203) 265-9535

**Georgia**

Norcross, GA  
(404) 441-0449

**Illinois**

Schaumburg, IL  
(708) 303-9900

**Kansas**

Lenexa, KS  
(913) 492-5406

**Maryland**

Columbia, MD  
(410) 290-3800

**Massachusetts**

Woburn, MA  
(617) 938-6200

**Minnesota**

Minneapolis, MN  
(612) 831-2666

**New Jersey**

Edison, NJ  
(908) 417-1000  
Mt. Laurel, NJ  
(609) 273-6420

**New Mexico**

Albuquerque, NM  
(505) 884-1900

**North Carolina**

Raleigh, NC  
(919) 790-8634

**Ohio**

Solon, OH  
(216) 248-1122

**Oklahoma**

Tulsa, OK  
(918) 663-2410

**Texas**

Austin, TX  
(512) 836-1341  
Carrollton, TX  
(214) 243-1600  
Houston, TX  
(713) 627-9800

**Utah**

Salt Lake City, UT  
(801) 972-5444

**Virginia**

Richmond, VA  
(804) 323-5510

**INTERNATIONAL  
REPRESENTATIVES  
& DISTRIBUTORS**
**Argentina**

*Semark S.A.*  
54-1-381-2108

**Australia**

*Dallas Identification*  
61-3-720-5344  
*Veltek Pty. Ltd.*  
61-3-9808-7511  
61-2-9713-4100

**Austria**

*Eurodis Electronics*  
43-1-610620

**Belgium**

*Acal Electronic Division*  
32-2-720-5983

**Bulgaria**

*Isomatic Lab Ltd.*  
359-2-731-463

**China**

*Waslin (H.K.) Limited*  
852-2-693-6811

**Czech Republic**

*HT-Eurep Elec.spol.sr.o.*  
42-2-663-13053

**Denmark**

*Micronor*  
45-86-81-65-22

**Finland**

*Dalma Electronics Oy*  
358-0271-1800

**France**

*Avnet EMG SA.*  
33-1-4965-2500  
*Rep'Tronic*  
(Representative)  
33-1-6013-9300

**Germany**

*Atlantik Elektronik*  
49-89-857-0000  
*Future Electronics*  
49-899-57270

**Great Britain**

*D.T. Electronics Ltd.*  
44-01203-466500  
*Farnell Dialogue*  
44-1-279-442971  
*Future Electronics Ltd.*  
44-1-753-687000  
*Joseph Electronics*  
44-21-643-6888  
*Silver Birch Mktg. Ltd.*  
44-1933-412285

**Greece**

*Drogeta Engineering*  
30-1-881-0948

**Hong Kong**

*Cet Ltd.*  
852-2-485-3899

**Hungary**

*HT-Eurep Elec.kft.*  
36-1-155-4748

**India**

*Hynetic Electronics*  
91-80-620-852  
*Zetex PLC*  
44-161-627-4963

**Ireland**

*Bloomer Elec. Ltd.*  
44-762-339818

**Israel**

*STG International*  
972-3-696-5231

**Italy**

*Comprel, S.P.A.*  
39-3-6255-3991  
*Comprel Rep S.R.L.*  
39-362-52 4941

**Japan**

*Microtek Inc.*  
81-3-5300-5535  
*Systems Marketing*  
81-33-254-2751

**Korea**

*Amerix Corporation*  
82-2-423-9623  
*Sangsoo Electronics Co.*  
82-2-780-5360

**Lithuania, Latvia & Estonia**

*Rytter*  
370-7-206244

**Malaysia**

*Dynamar*  
60-4-228-1860

**The Netherlands**

*Alcom Electronics BV*  
31-10-451-9533

**New Zealand**

*Components & Instrumentation NZ Ltd.*  
64-4-566-3222

**Norway**

*BIT Elektronikk A.S.*  
47-66-98-1370

**Poland**

*HT-Eurep Elec.sp.zo.o.*  
48-2-621-7704

**Portugal**

*Comelta, S.A.*  
351-1-941-6507

**Singapore**

*Dynamar*  
65-542-1878

**Slovenia & Croatia**

*Cadix d.o.o.*  
386-61-1592577

**South Africa**

*Tarsus Technology P.V.T.*  
27-11-886-3165

**Spain**

*Comelta, S.A. Barcelona*  
34-3-582-1991  
*Comelta, S.A. Madrid*  
34-1-327-0614

**Sweden**

*Dipcom Electronics AB*  
46-87-522-480

**Switzerland**

*Computer Controls AG*  
41-1-308-6666

**Taiwan**

*Landcol Enterprises*  
886-2-377-6070

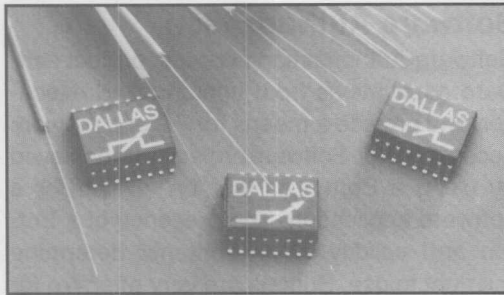
**Thailand**

*Dynamar*  
66-2-376-0132

**Turkey**

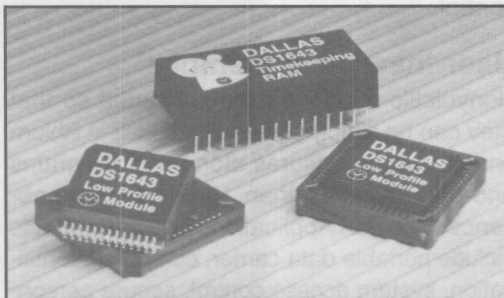
*Politeknik Elektronik*  
90-232-232-7432





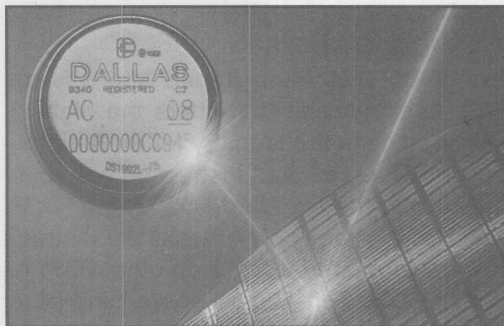
### Silicon Timed Circuits

Silicon Timed Circuits (often referred to as delay lines) are chips that make subtle adjustments to the timing of high-performance electronics so that they will perform optimally. Because of the precision that lasers provide, some Silicon Timed Circuits can make timing adjustments down to a fraction of a nanosecond, which is the time it takes light to travel about a foot. For more information, call our Timing Problem Hotline at (214) 450-5348.



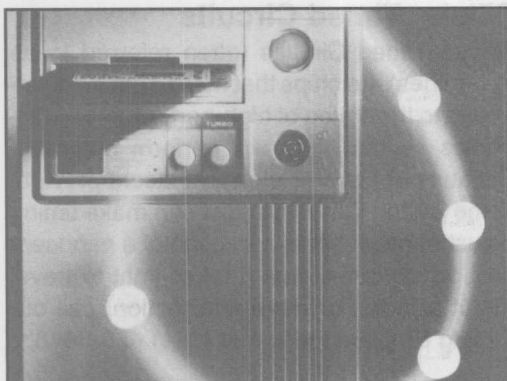
### Timekeeping

A self-contained lithium energy source in conjunction with a silicon chip and quartz form a permanently powered clock/calendar within a single component. Various computer interfaces are available including phantom, serial, PC DOS, and byte-wide memory.



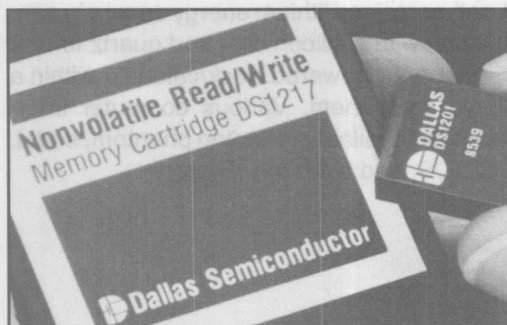
### Automatic Identification

Touch Memory™ is a self-stick, Silicon Label™ in a stainless steel can. This MicroCan™ provides all the advantages canning has to offer, such as low cost, ruggedness, and the ability to preserve contents. The MicroCan's greatest advantage, however, is that a standalone chip can leave the confines of the computer and travel virtually anywhere to bring digital data to the point of use. Information can be updated time after time while the label is still affixed to its object. Wherever the silicon-labelled object goes, information is served up on the spot without recourse to remote networks. This family also includes low-cost memory chips in T0-92 packages.



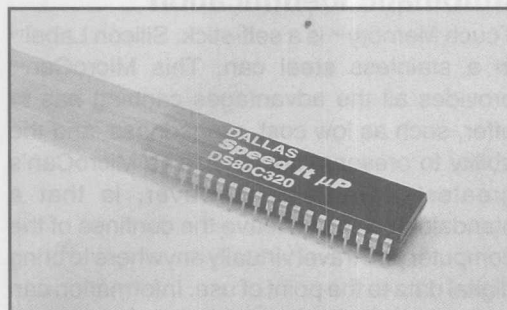
## Software Authorization

Software Authorization products protect software applications from unauthorized execution and provide a means for PC and network access control. Software protection is achieved by using a Button as the "on" switch for a software application. The presence of a Button and validity of its contents determine the right to use. Buttons are very effective for implementing time- or count-based metering as a way of extending the temporary right to use software while maintaining protection control.



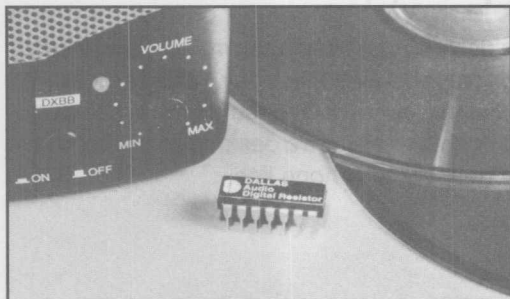
## User-Insertable Memory

Nonvolatile memories are packaged so that they can be simply plugged in. A built-in lithium energy source ensures storage of programs and data for more than 10 years in the absence of power. Applications for such products include portable data carrier, computer identification, system access control, secure personnel areas, calibration, automatic system setup, and traveling work records. All products can be read or written by a PC.



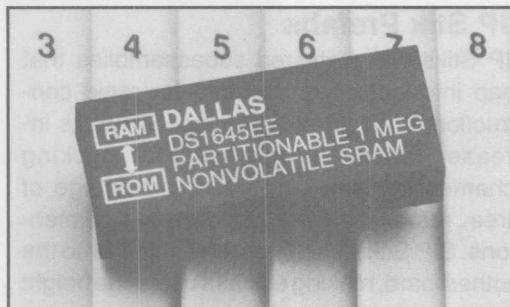
## Microcontrollers

The DS80C320 High-Speed Micro is an 8051 family device that offers the highest performance in the industry for an 8-bit microcontroller. Pin- and instruction set-compatible with the standard 80C32, the High-Speed Micro uses only 4 clocks per instruction, as compared with 12 on other 8051's. Our DS500x Soft Micros convert industry-standard byte-wide SRAM into high-performance, nonvolatile read/write storage.



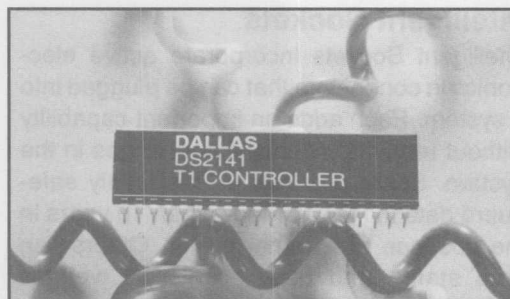
## System Extension

These products add a variety of special features to systems without encumbering design. A digital potentiometer is an all-silicon version of an electrical element used in almost all electronic equipment. CPU supervisors monitor vital conditions for a microprocessor. Digital Thermometers measure temperature and directly output a digital number.



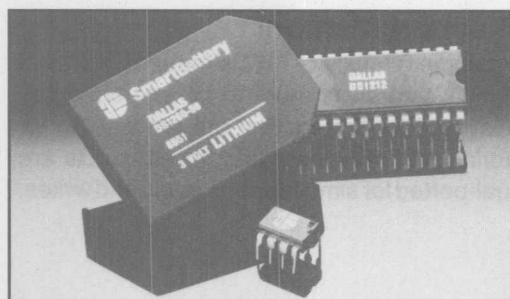
## Nonvolatile RAM

Dallas Semiconductor has combined its circuitry and understanding of ultra low-power CMOS SRAM with improvements in long-life lithium power sources to develop a family of nonvolatile RAMs that retain data for more than 10 years in the absence of main power.



## Telecommunications

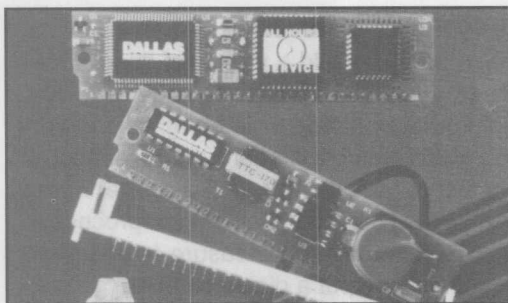
A comprehensive product family addresses the requirements of high-speed, digital voice/data transmission and monitoring in T1, CEPT, or Primary Rate ISDN networks. The DS2151/53 T1/E1 Single-Chip Transceivers combine all the circuitry needed to connect to a T1 or E1 line in a single package.



## Battery Backup & Chargers

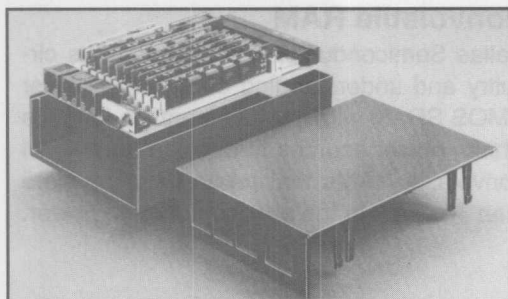
The Battery Backup chip set crashproofs microprocessor-based systems, ensuring that no information is lost when main power fails. When power returns, computing resumes as if the failure had not occurred. Battery Chargers contain all the circuitry needed to recharge a 3-cell NiCad or lithium battery pack in a 3-pin package.





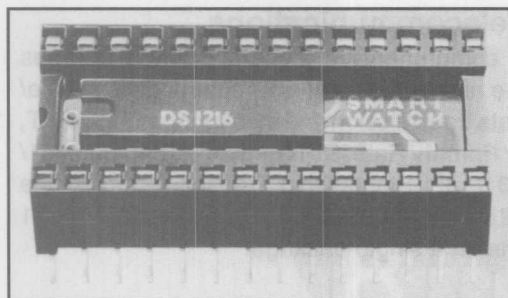
## Teleservicing

Teleservicing products can monitor equipment performance 24 hours a day, release software revisions, perform diagnostics, and make adjustments — all from a desktop computer over an ordinary telephone line.



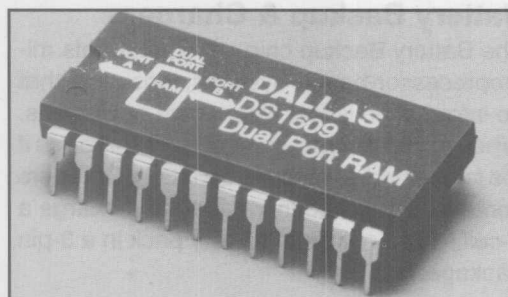
## SIP Stik Prefabs

SIP Stiks are pretested subassemblies that snap into locking connectors for rapid construction of electronic systems. SIP Stiks increase density over traditional packing schemes five times by taking advantage of three, rather than the standard two, dimensions. SIP Stiks insert perpendicularly into the motherboard, making efficient use of the height dimension.



## Intelligent Sockets

Intelligent Sockets incorporate active electronics in connectors that can be plugged into a system. Each adds an important capability without requiring substantive changes in the system. Some products in this family safeguard data in RAM for more than 10 years in the absence of external power. Others can time stamp and date events as well as nonvolatize RAM.



## Multiport Memory

A complete family of FIFOs features identical pinouts that allow them to be interchanged. Designed for first-in, first-out processing for storing and retrieving data, the products are dual-ported for simultaneous reads and writes.

Dallas Semiconductor Corporation designs, manufactures, and markets electronic chips and chip-based subsystems. Founded in 1984, the Company uses customer problems as an entry point to develop products with wide-spread applications. The Company is committed to new product development as a means to increase future revenues and to diversify its markets, products, and customers.

Advanced technologies have given the Company a competitive edge over traditional approaches to semiconductors. Combining lithium energy cells with low-power CMOS chips powers chips for the useful life of the equipment. Direct laser writing enhances chip capabilities with high levels of precision and unique identities.

In its 11-year history, Dallas Semiconductor has developed 215 base products with over 1,000 variations shipped to more than 8,000 customers worldwide. A direct sales force and distribution network sell to original equipment manufacturers (OEMs) in personal computers and workstations, scientific and medical equipment, industrial controls, automatic identification, telecommunications, consumer electronics, and other markets.

Sales for 1994 totaled \$181,432,000. Dallas Semiconductor has 850 employees. On March 19, 1990, the Company started trading on the New York Stock Exchange under the symbol DS.

## TECHNOLOGY

Dallas Semiconductor's special technologies make possible Soft Silicon™ solutions—dynamic, flexible, chip-based products that can be molded in the final manufacturing stages or during use. Soft Silicon is made possible by lithium energy and direct laser writing.

## Lithium

Using micro energy management techniques, Dallas Semiconductor has reduced power requirements to the point where a miniature lithium energy source powers products for the useful life of the equipment. Chips and Stiks (snap-in subassemblies) are made virtually crashproof with minimum current design techniques and special freshness seals that keep lithium cells from expending any energy until power is applied for the first time. Through these technologies, Dallas products remember data throughout their operating life and can accept change.

## Laser

Direct laser writing makes each chip unique at low cost. A sub-micron positioning laser and control software developed at Dallas can engrave individual chips with digital patterns. This ability to routinely alter, reconfigure, or program individual chips after completion of wafer fabrication broadens the application base of products having similar design. Direct laser writing allows Dallas Semiconductor to develop highly accurate products for applications where precision is paramount.

As a result of these Late Definition technologies, exact chip definition can be left to the OEM. Certain chips can even be defined and redefined by the end system itself.

## MANUFACTURING AND FACILITIES

As of January 1, 1995, the Company owns 342,500 square feet of building space and 22.9 acres of land in Dallas. The Company's wafer fabrication facility is a model of efficiency. In order to add capacity for growth the Company built a new advanced wafer fabrication facility that began production in 1994. The new fab is an important asset in terms of its capacity and process capabilities.

## QUALITY SYSTEM

Product quality at Dallas Semiconductor results from a combination of design techniques, vendor controls, manufacturing methods, process monitors, and quality control inspections. SPC monitors placed at strategic points ensure that potential defects are detected promptly.

## QUALITY CONTROL PROCESSES

- **Incoming Quality Control (IQC):** Piece parts and raw materials are inspected by IQC. New vendors and piece parts receive a First Article Inspection; subsequent incoming materials receive a sample inspection per MIL-STD-105.
- **In-Process Inspections:** Each manufacturing operation inspects its own work, ensuring immediate feedback and preventing deviations from going undetected due to subsequent processing.
- **Statistical Process Control (SPC):** Implemented in manufacturing, this process determines what inputs to the product flow are critical and how to track and control those inputs. Quality Engineering provides training, computer analysis, and feedback to manufacturing.
- **In-Process Sample Tests:** In order to guarantee the accuracy and completeness of in-process inspections and SPC monitors, QC Toll Gates at strategic locations perform sample inspections per MIL-STD-105.

## RELIABILITY SYSTEM

Reliability is accomplished through a rigorous, comprehensive methodology of qualifying, analyzing, and monitoring new equipment, processes, products, and packages. A state-of-the-art environmental facility allows accelerated stresses to be performed and monitored in-house. In addition, a metallurgical laboratory has been equipped to perform real-time x-ray, x-ray fluorescence, and solderability measurements.

To minimize the human influence on the outcome of the reliability activity, a dedicated group of technicians and assistants handle all reliability stressing and testing. Reliability data resides on a customized computer-based tracking and retrieval system. Technical support includes oven and chamber calibrations, 100% electrical board checks, and strict electrostatic protection.

## PRODUCT QUALIFICATION

Product qualification activity at Dallas Semiconductor involves a series of accelerated stress tests applied to production-ready material and follows a defined qualification plan. Random samples from at least three production lots, equally representing the production version of the product, are tested to meet reliability requirements. Any device failures detected during production qualification or subsequent monitoring are fully analyzed in our Failure Analysis Laboratory.

Products at Dallas Semiconductor fall into one of three classifications: Prototype or Engineering Sample, Prequal, and Fully Qualified.

- **Prototype or Engineering Sample:** Prototype products have not been fully characterized to all data sheet limits. However, based upon limited data, these products will meet data sheet limits. Final test and all processes used to manufacture the product are under engineering control. Qualification of the product has not started. The brand on prototype products will be PROTO or ES.
- **Prequal:** Prequal products meet prototype requirements and are characterized to all data sheet limits. Final test and all processes used to manufacture the product are stable and under manufacturing control. Qualification of the product has started.
- **Fully Qualified:** Fully qualified products meet prototype and prequal requirements. The qualification requirements given in the next section have been completed. Product must statistically meet reliability failure rates and quality requirements as established by Quality and Reliability Engineering.

Tables 1, 2 and 3 list the tests which a Dallas Semiconductor product must pass in order to be classified as fully qualified.

## RELIABILITY MONITOR PROGRAM

In order to maintain continuous qualification status on all products, Dallas Semiconductor has implemented an extensive Reliability Monitor Program (RMP). The RMP monitors all design, wafer fabrication, and assembly processes in the qualified products database. Product is selected monthly from finished goods and subjected to a series of reliability tests similar to those used in the original qualification. Any failures generated from these tests require analysis to root cause and corrective action.

Data from the RMP is published quarterly and is available on demand.

FULL QUALIFICATION REQUIREMENTS FOR INTEGRATED CIRCUIT PRODUCTS Table 1

STRESS/TEST	CONDITION	DURATION	ACCEPTANCE CRITERIA (LTPD)
Outgoing Elec. Test	Data Sheet	0 Hr.	0.15%
Infant Life	125°C, 7.0V	48 Hr.	0.3%
High Temperature Operating Life	125°C, 5.5V	1000 Hr.	*0.4%
Use Condition Prediction	55°C, 5.5V	10 years	50 Fits
High Voltage Life	125°C, 7.0V	1000 Hr.	*0.4%
High Temperature Storage	150°C, No Bias	1000 Hr.	2.0%
Temperature Humidity Bias	85°C/85% RH, 5.5V	1000 Hr.	1.0%
Autoclave	121°C, 2 ATM Steam, Unbiased	168 Hr.	1.5%
Temperature Cycle	-55°C to +125°C	1000 cycle	1.0%
X-Ray	MIL-STD-883 Method 2012		15%
Bond Pull	MIL-STD-883 Method 2011	Premold	1.5%
Dimensions	MIL-STD-883 Method 2016		15%
Lead Integrity	MIL-STD-883 Method 2004		3.0%
Solderability	MIL-STD-883 Method 2003	8 Hr. Steamage	3.0%
ESD	MIL-STD-883 Method 3015		> ±1000 volts
Latch-up	JEDEC Std. 17		> 100 mW/pin

\* Combined high voltage life and operating life requirement.



FULL QUALIFICATION REQUIREMENTS FOR MODULE PRODUCTS Table 2

STRESS/TEST	CONDITION	DURATION	ACCEPTANCE CRITERIA (LTPD)
Outgoing Elec. Test	Data Sheet	0 Hr.	0.15%
Use Condition Prediction	55°C, 5.5V	10 years	50 Fits
High Temperature Storage	85°C, No Bias	1000 Hr.	2.0%
*Temperature Humidity Bias	85°C/85% RH, 5.5V	959 Hr.	1.0%
Temperature Cycle	-40°C to +85°C	1000 cycle	1.0%
X-Ray	MIL-STD-883 Method 2012		15%
Dimensions	MIL-STD-883 Method 2016		15%
Lead Integrity	MIL-STD-883 Method 2004		3.0%
Solderability	MIL-STD-883 Method 2003	8 Hr. Steamage	3.0%

\* Selected products.

FULL QUALIFICATION REQUIREMENTS FOR SIPSTIK AND TOUCH MEMORY PRODUCTS Table 3

STRESS/TEST	CONDITION	DURATION	ACCEPTANCE CRITERIA (LTPD)
Outgoing Elec. Test	Data Sheet	0 Hr.	0.15%
High Temperature Storage	85°C, No Bias	1000 Hr.	7.0%
Temperature Humidity	60°C/90% RH	288 Hr.	7.0%
Temperature Cycle	-40°C to +85°C	500 cycle	7.0%

## LIBRIS GUIDA

## USER'S GUIDE

## SECTION 1: INTRODUCTION

The Dallas Semiconductor High-Speed Microcontroller is an 8051-compatible device that provides improved performance and power consumption compared to the original version. It retains instruction set and object code compatibility with the 8051, yet performs the same operations in fewer clock cycles. Consequently, more throughput is possible for the same crystal speed. As an alternative, the High-Speed Microcontroller can be run slowly to save power. The more efficient design allows a much slower crystal speed to get the same results as an original 8051, using much less power.

The fundamental innovation of the High-Speed Microcontroller is the use of only four clocks per instruction cycle compared with twelve for the original 8051. This results in up to 3 times improvement in performance. In addition, the High-Speed Microcontroller is updated with several new peripherals and features while providing all of the standard features of an 80C32. These include 256 bytes of on-chip RAM, 32 I/O ports, three 16-bit timer/counters, and an on-chip UART. All devices provide 256 bytes of RAM for variables and stack. 128 bytes can be reached using direct addressing and 128 using indirect addressing. The RAM area matches the standard 80C32.

In addition to improved efficiency, many members of the High-Speed Microcontroller family can operate at a maximum clock rate of 33 MHz. Combined with the 3 times performance, this allows for a maximum performance equivalent to a 99 MHz 8051. This level of computing power is comparable to many 16-bit processors, but without the added expense and complexity of implementing a 16-bit interface.

A number of peripherals were added to the original 80C32 core when designing the High-Speed Microcontroller family. Some devices have a programmable Watchdog Timer to supervise the system. It will count up to a user programmable interval and then reset the CPU unless cleared by software. Other features such as a second, full-function UART and dual data pointers are available to minimize external hardware and system complexity. The incorporation of 6 external interrupts allows greater flexibility in dealing with external events.

Some members of the High-Speed Microcontroller family incorporate Power Management Modes which allow the device to dynamically vary the internal clock speed from 4 clocks per cycle (default) to 64 or 1024 clocks per cycle. Because power consumption is directly proportional to clock speed, the device can reduce its operating frequency during periods of little or no activity. This greatly reduces power consumption. The switch-back feature allows the device to quickly return to divide by 4 mode upon receipt of an interrupt or serial port activity, allowing the device to respond to external events while in Power Management Mode.

Various memory configurations are available with the High-Speed Microcontroller family. EPROM and Mask programmable ROM versions are available for program memory. Some versions incorporate extended MOVX SRAM on-chip, reducing or eliminating the need for external data memory. This memory can be made nonvolatile in the DS87C530 through the use of an external lithium battery.

**SECTION 2: ORDERING INFORMATION**

The High-Speed Microcontroller family follows the part numbering convention shown below. Note that all com-

binations of devices are not currently available. Please refer to individual data sheets for the available versions.

**DS80C320-MCG**

<b>SPEED:</b>	G 25 MHz
	L 33 MHz
<b>TEMPERATURE:</b>	C 0°C to 70°C
	N -40°C to +85°C
<b>PACKAGE:</b>	M PLASTIC DIP
	Q PLCC
	E THIN PLASTIC QUAD FLAT PACK (TQFP)
	W 40-PIN WINDOWED CERDIP
	K 52-PIN WINDOWED CERQUAD
<b>OPERATING VOLTAGE:</b>	0 +5V
	3 +3V OR WIDE VOLTAGE
<b>MEMORY TYPE:</b>	0 ROMless
	3 ROM
	7 EPROM

The standard High-Speed Microcontroller offers four 8-bit I/O ports. ROMless versions use Port 0 and Port 1 as address and data buses. In those versions, only two ports are available for general purpose I/O. Each I/O port is a Special Function Register that can be written or read. The I/O port has a latch that retains the value which software writes. In general, during a read operation, software reads the state of the external pin. Each port is represented by an SFR location.

Three 16-bit Timers/Counters are available in the High-Speed Microcontroller. Each timer is contained in two SFR locations that can be written or read by software. The timers are controlled by other SFRs described in Section 4.

The High-Speed Microcontroller provides one or two UARTs. These are controlled and accessed as SFRs. Each UART has an address that is used to read or write the UART. The same address is used for both read and write operations. The microcontroller distinguishes between a read and a write by the instruction. Each UART is controlled by its own SFR control register.

All peripherals and registers that are not explicit instructions in the High-Speed Microcontroller are controlled via Special Function Registers (SFRs). All SFRs are described in Section 4. The most commonly used registers that are basic to the architecture are also described below.

**Accumulator**  
The Accumulator is the primary register used in the High-Speed Microcontroller. It is the source and destination of most math, data movement, decision, and other operations. Although it can be bypassed, most high-speed instructions require the use of the Accumulator (ACC) as one argument.

**B Register**  
The B register is used as the second 8-bit argument in multiply and divide operations. When not used for these purposes, the B register can be used as a general purpose register.

**Program Status Word**  
The Program Status Word holds a selection of bit flags that include the Carry Flag, Auxiliary Carry Flag, Zero



### SECTION 3: ARCHITECTURE

The High-Speed Microcontroller is based on the industry standard 80C52. The core is an accumulator based architecture using internal registers for data storage and peripheral control. It executes the standard 8051 instruction set. This section provides a brief description of each architecture feature. Details concerning the programming model, instruction set, and register descriptions are provided in Section 4.

#### ALU

The ALU is responsible for math functions, comparisons, and general decision making in the High-Speed Microcontroller. The ALU is not explicitly used by software. Instruction decoding prepares the ALU automatically and passes it the appropriate data. The ALU primarily uses two special function registers (SFRs) as the source and destination for all operations. These are the Accumulator and B register. The ALU also provides status information in the Program Status Register. The SFRs are described below.

#### SPECIAL FUNCTION REGISTERS

All peripherals and operations that are not explicit instructions in the High-Speed Microcontroller are controlled via Special Function Registers (SFRs). All SFRs are described in Section 4. The most commonly used registers that are basic to the architecture are also described below.

#### Accumulator

The Accumulator is the primary register used in the High-Speed Microcontroller. It is the source and destination of most math, data movement, decisions, and other operations. Although it can be bypassed, most high-speed instructions require the use of the Accumulator (ACC) as one argument.

#### B Register

The B register is used as the second 8-bit argument in multiply and divide operations. When not used for these purposes, the B register can be used as a general purpose register.

#### Program Status Word

The Program Status Word holds a selection of bit flags that include the Carry Flag, Auxiliary Carry Flag, Gen-

eral Purpose Flag, Register Bank Select, Overflow Flag, and Parity Flag.

#### Data Pointer(s)

The Data Pointer is used to designate a memory address for the MOVX instruction. This address can point to a MOVX RAM location, either on- or off-chip, or a memory mapped peripheral. When moving data from one memory area to another or from memory to a memory mapped peripheral, a pointer is needed for both the source and destination. Thus, the High-Speed Microcontroller offers two data pointers. The user selects the active pointer via a dedicated SFR bit.

#### Stack Pointer

The High-Speed Microcontroller provides a Stack in the scratchpad RAM area discussed below. The Stack Pointer denotes the register location at the top of the Stack, which is the last used value. The user can place the Stack anywhere in scratchpad RAM by setting the Stack Pointer to that location.

#### I/O Ports

The standard High-Speed Microcontroller offers four 8-bit I/O ports. ROMless versions use Port 0 and Port 2 as address and data busses. In those versions, only two ports are available for general purpose I/O. Each I/O port is a Special Function Register that can be written or read. The I/O port has a latch that retains the value which software writes. In general, during a read operation, software reads the state of the external pin. Each port is represented by an SFR location.

#### Timer/Counters

Three 16-bit Timer/Counters are available in the High-Speed Microcontroller. Each timer is contained in two SFR locations that can be written or read by software. The timers are controlled by other SFRs described in Section 4.

#### UARTs

The High-Speed Microcontroller provides one or two UARTs. These are controlled and accessed as SFRs. Each UART has an address that is used to read or write the UART. The same address is used for both read and write operations. The microcontroller distinguishes between a read and a write by the instruction. Each UART is controlled by its own SFR control register.

### SCRATCHPAD REGISTERS (RAM)

The High-Speed Core provides 256 bytes of Scratchpad RAM for general purpose data and variable storage. The first 128 bytes are directly available to software. The second 128 are available through indirect addressing discussed below. Selected portions of this RAM have other optional functions.

### Stack

The stack is a RAM area that the microcontroller uses to store return address information during Calls and Interrupts. The user can also place variables on the stack when necessary. The Stack Pointer mentioned above designates the RAM location that is the top of the stack. Thus, depending on the value of the Stack Pointer, the stack can be located anywhere in the 256 bytes of RAM. A common location would be in the upper 128 bytes of RAM, as these are accessible through indirect addressing only.

### Working Registers

The first thirty two bytes of the Scratchpad RAM can be used as four banks of eight Working Registers for high speed data movement. Using four banks, software can quickly change context by simply changing to a different bank. In addition to the Accumulator, the Working Registers are commonly used as a data source or destination. Some of the Working Registers can also be used as pointers to other RAM locations (indirect addressing).

### PROGRAM COUNTER

The Program Counter (PC) is a 16-bit value that designates the next program address to be fetched. On-chip hardware automatically increments the PC value to move to the next ROM location.

### ADDRESS/DATA BUS

The High-Speed Microcontroller addresses a 64KB program and 64KB data memory area. In the ROMless versions, all memory is outside. Other versions use a combination of internal and external memory. When external memory is accessed, Ports 0 and 2 are used as a multiplexed address and data bus. Port 2 provides the address MSB. Even versions with internal memory can use the bus on Ports 0 and 2 to access more memory.

### WATCHDOG TIMER

The Watchdog Timer provides a supervisory function for applications that cannot afford to run out of control. The Watchdog Timer is a programmable free running timer. If allowed to reach the termination of its count, if enabled, the Watchdog will reset the CPU. Software must prevent this by clearing or resetting the Watchdog prior to its time-out.

### POWER MONITOR

Some members of the High-Speed Microcontroller family incorporate a band-gap reference and analog circuitry to monitor the power supply conditions. If  $V_{CC}$  begins to drop out of tolerance, the Power Monitor will issue an optional early warning Power-fail Interrupt. If power continues to fall, the Power Monitor will invoke a reset condition. This will remain until power returns to normal operating voltage. The Power Monitor also functions on power up, holding the microcontroller in a reset state until power is stable.

### INTERRUPTS

The High-Speed Microcontroller is capable of evaluating a number of interrupt sources simultaneously. Each version of the High-Speed Microcontroller provides a different number of interrupt sources. Each interrupt has an associated interrupt vector, flag, priority, and enable. These interrupts can be globally enabled or disabled.

### TIMING CONTROL

The High-Speed Microcontroller provides an on-chip oscillator for use with an external crystal. This can be bypassed by injecting a clock source into the XTAL1 pin. The clock source is used to create machine cycle timing (four clocks), ALE,  $\overline{PSEN}$ , Watchdog, Timer, and serial baud rate timing. In addition, some devices incorporate an on-chip ring oscillator which can be used to provide an approximately 2–4 MHz clock source.

### REAL-TIME CLOCK

The DS87C530 incorporates a real-time clock (RTC), which is accessed via SFR locations. The RTC is divided into hour, minute, second, and subsecond registers, and also incorporates a 65536 day calendar. Alarm registers allow the RTC to issue interrupts at a specific time once per day, or as a recurring alarm every hour, minute, or second. An external watch crystal and lithium power source allow the processor to maintain timekeeping in the absence of  $V_{CC}$ .

FEATURE SUMMARY

The High-Speed Microcontroller family offers a combination of features and peripherals as shown in Table 3-1. This User's Guide is designed as a comprehensive

guide covering all the features available in the High-Speed Microcontroller family. The designer should investigate the specific data sheet to determine which features are available on a particular device.

PRODUCT FEATURE MATRIX 3-1

FEATURE	DS80C310	DS80C320	DS80C323	DS83C520	DS87C520	DS87C530
Internal Program ROM				16KB Mask ROM	16KB EPROM	16KB EPROM
Internal Scratchpad RAM	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Internal MOVX SRAM				1KB SRAM	1KB SRAM	1KB SRAM
Serial Ports	1	2	2	2	2	2
External Interrupts	6	6	6	6	6	6
16-bit Timers	3	3	3	3	3	3
Watchdog Timer		✓	✓	✓	✓	✓
Power-fail/Precision Reset		✓	✓	✓	✓	✓
Power-Fail Interrupt		✓	✓	✓	✓	✓
Data Pointers	2	2	2	2	2	2
Power Management Modes				✓	✓	✓
Ring Oscillator		✓	✓	✓	✓	✓
EMI Reduction Mode				✓	✓	✓
Real-Time Clock						✓
Nonvolatile SRAM						✓
Operating Voltage	4.5V-5.5V	4.5V-5.5V	2.7V-5.5	4.5V-5.5V	4.5V-5.5V	4.5V-5.5V

oscillator for use with an external crystal. This can be bypassed by injecting a clock source into the XTAL1 pin. The clock source is used to create machine cycle timing (low clock), ALE, PSEN, Watchdog, Timer, and serial bus timing. In addition, some devices incorporate an on-chip ring oscillator which can be used to provide an approximately 2-4 MHz clock source.

**REAL-TIME CLOCK**  
The DS87C520 incorporates a real-time clock (RTC) which is accessed via SPI locations. The RTC is divided into hour, minute, second, and subsecond registers and also contains a 32-bit day calendar. Alarm registers allow the RTC to issue interrupts at a specific time once per day, or as a recurring alarm every hour, minute, or second. An external watch crystal and 10K pull-up source allow the processor to maintain timekeeping in the absence of VCC.

The Program Counter (PC) is a 16-bit value that designates the next program address to be fetched. On-chip hardware automatically increments the PC value to move to the next ROM location.

**ADDRESS/DATA BUS**  
The High-Speed Microcontroller addresses a 64KB program and 64KB data memory area. In the ROMless versions, all memory is outside. Other versions use a combination of internal and external memory. When external memory is accessed, Ports 0 and 2 are used as a multiplexed address and data bus. Port 2 provides the address MSB. Even versions with internal memory use the bus on Port 0 and 2 to access more memory.

## SECTION 4: PROGRAMMING MODEL

This section provides a programmer's overview of the High-Speed Microcontroller core. It includes information on the memory map, on-chip RAM, Special Function Registers (SFRs), and instruction set. The programming model of the High-Speed Microcontroller is very similar to that of the industry standard 80C52. The memory map is identical. It uses the same instruction set, though instruction timing is improved. Several new SFRs have been added.

### MEMORY ORGANIZATION

The High-Speed Microcontroller, like the 8052, uses several distinct memory areas. These are Registers, program memory, and data memory. Registers serve to control on-chip peripherals and as RAM. Note that Registers (on-chip RAM) are separate from data memory. Registers are divided into three categories including directly addressed on-chip RAM, indirectly addressed on-chip RAM, and Special Function Registers. The program and data memory areas are discussed under Memory Map. The Registers are discussed under Register Map.

### MEMORY MAP

The High-Speed Microcontroller uses a memory addressing scheme that separates program memory (ROM) from data memory (RAM). Each area is 64KB beginning at address 0000h and ending at FFFFh as shown in Figure 4-1. The program and data segments can overlap since they are accessed in different ways. Program memory is fetched by the microcontroller automatically. These addresses are never written by software. In fact, there are no instructions that allow the ROM area to be written. There is one instruction (MOVC) that is used to explicitly read the program area. This is commonly used to read look-up tables. The data memory area is accessed explicitly using the MOVX instruction. This instruction provides multiple ways of specifying the target address. It is used to access the 64KB of data memory.

The address and data range of devices with on-chip program and data memory overlap the 64K memory space. When on-chip memory is enabled, accessing memory in the on-chip range will cause the device to access internal memory. Memory accesses beyond the internal range will be addressed externally via ports 0 and 2.

The ROMSIZE feature allows software to dynamically configure the maximum address of on-chip program memory. This allows the device to act as a bootstrap loader for an external Flash or nonvolatile SRAM. Secondly, this method can also be used to increase the amount of available program memory from 64KB to 80KB without bank switching. For more information on this feature, please consult Section 6.

Program and data memory can also be increased beyond the 64KB limit using bank switching techniques. This is described in Application Note 81, Memory Expansion with the High-Speed Microcontroller family.

### REGISTER MAP

The Register Map is illustrated in Figure 4-2. It is entirely separate from the program and data memory areas mentioned above. A separate class of instructions is used to access the registers. There are 256 potential register location values. In practice, the High-Speed Microcontroller has 256 bytes of Scratchpad RAM and up to 128 Special Function Registers (SFRs). This is possible since the upper 128 Scratchpad RAM locations can only be accessed indirectly. That is, the contents of a Working Register (described below) will designate the RAM location. Thus a direct reference to one of the upper 128 locations must be an SFR access. Direct RAM is reached at locations 0 to 7Fh (0 to 127). SFRs are accessed directly between 80h and FFh (128 to 255). The RAM locations between 128 and 255 can be reached through an indirect reference to those locations.

Scratchpad RAM is available for general purpose data storage. It is commonly used in place of off-chip RAM when the total data contents are small. When off-chip RAM is still needed, the Scratchpad area will still provide the fastest general purpose access. Within the 256 bytes of RAM, there are several special purpose areas. These are described as follows.

### Bit Addressable Locations

In addition to direct register access, some individual bits are also accessible. There are individually addressable bits in both the RAM and SFR area. In the Scratchpad RAM area, registers 20h to 2Fh are bit addressable. This provides 128 ( $16 * 8$ ) individual bits available to software. A bit access is distinguished from a full register access by the type of instruction. Addressing modes



As part of the lower 128 bytes of RAM, there are four banks of eight Working Registers (each). The Working Registers are general purpose RAM locations that can be addressed in a special way. They are designated R0 through R7. Since there are four banks, the currently selected bank will be used by any instruction using R0–R7. This allows software to change context by simply switching banks. This is controlled via the Program Status Word register in the SFR area described below. The Working Registers also allow their contents to be used for indirect addressing of the upper 128 bytes of RAM. Thus an instruction can designate the value

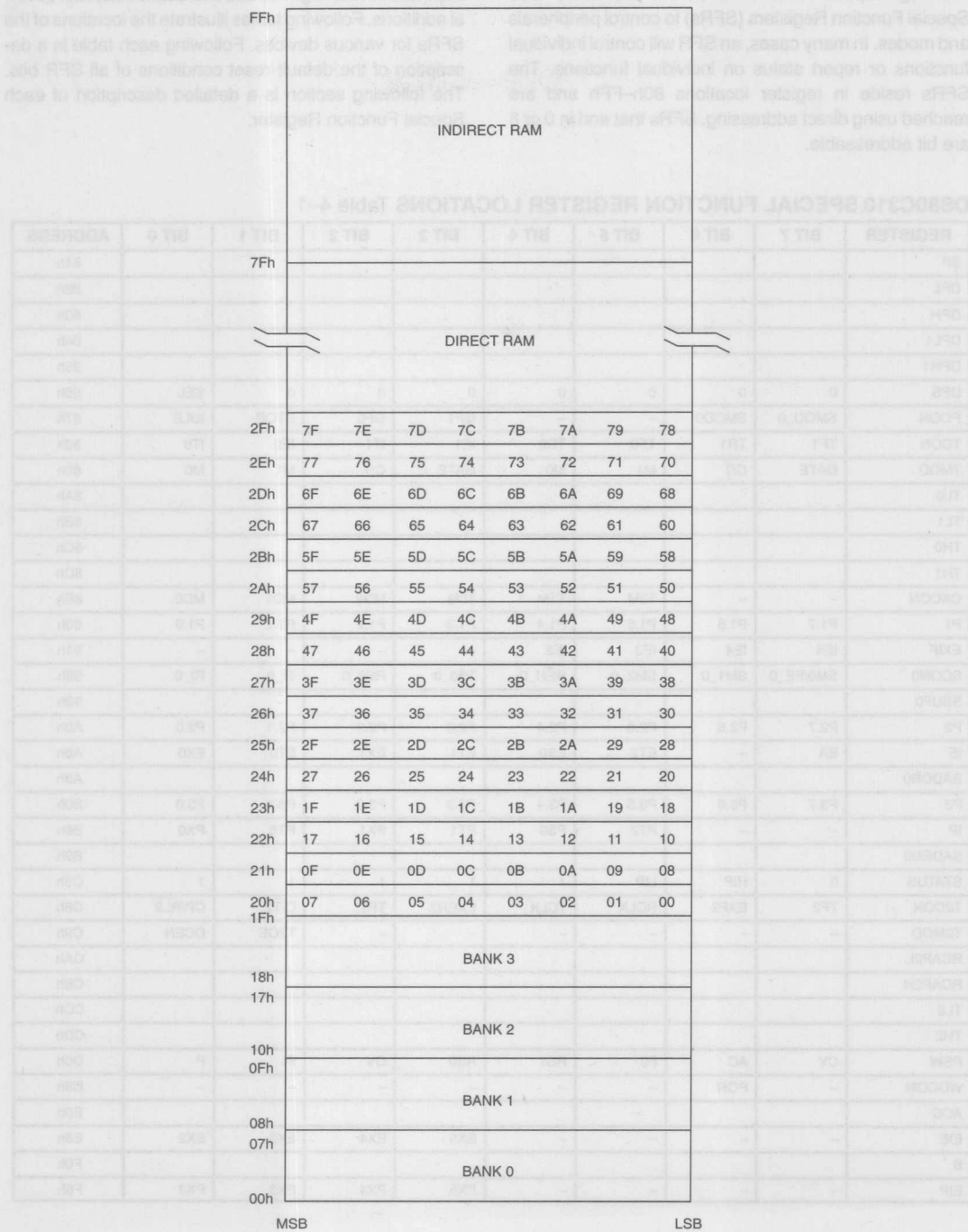
FFFFh	PROGRAM MEMORY
-------	----------------

FFh	INDIRECT RAM	12
7Fh	DIRECT RAM	12
00h		

## Stack

Another use of the Scratchpad area is for the programmer's stack. This area is selected using the Stack Pointer (SP,81h) SFR. Whenever a jump, call, or interrupt is invoked, the return address is placed on the Stack. It also is available to the programmer for variables, etc. Since the Stack can be moved, there is no fixed location within the RAM designated as Stack. The Stack Pointer will default to 07h on reset. The user can then move it as needed. A convenient location would be the upper RAM area (>7Fh) since this is only available indirectly. The SP will point to the last used value. Therefore, the next value placed on the Stack is put at SP + 1. Each PUSH or CALL will increment the SP by the appropriate value. Each POP or RET will decrement as well.

# **SCRATCHPAD REGISTER ADDRESSING Figure 4-3**



**SPECIAL FUNCTION REGISTERS**

The High-Speed Microcontroller, like the 8051, uses Special Function Registers (SFRs) to control peripherals and modes. In many cases, an SFR will control individual functions or report status on individual functions. The SFRs reside in register locations 80h–FFh and are reached using direct addressing. SFRs that end in 0 or 8 are bit addressable.

All standard SFR locations from the original 8051 are duplicated in the High-Speed Microcontroller, with several additions. Following tables illustrate the locations of the SFRs for various devices. Following each table is a description of the default reset conditions of all SFR bits. The following section is a detailed description of each Special Function Register.

**DS80C310 SPECIAL FUNCTION REGISTER LOCATIONS** Table 4–1

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	–	–	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	–	–	–	–	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	–	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	–	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
STATUS	0	HIP	LIP	1	1	1	1	1	C5h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	–	POR	–	–	–	–	–	–	D8h
ACC									E0h
EIE	–	–	–	–	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	–	PX5	PX4	PX3	PX2	F8h

**DS80C310 SPECIAL FUNCTION REGISTER RESET VALUES Table 4-2**

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	0	0	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	0	0	0	0	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
P3	1	1	1	1	1	1	1	1	B0h
IP	0	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
STATUS	0	0	0	1	1	1	1	1	C5h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	0	0	0	0	0	0	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	0	0	0	0	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	0	0	0	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	0	0	0	0	0	0	0	0	F8h



DS80C320/DS80C323 SPECIAL FUNCTION REGISTER LOCATIONS Table 4-3

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	–	RGMD	RGSL	BGS	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TL_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TL_1	RI_1	C0h
SBUF1									C1h
STATUS	PIP	HIP	LIP	1	1	1	1	1	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
ACC									E0h
EIE	–	–	–	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	PWDI	PX5	PX4	PX3	PX2	F8h

Shaded bits are Timed Access protected.

**DS80C320/DS80C323 SPECIAL FUNCTION REGISTER RESET VALUES** Table 4-4

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	—	—	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	—	0	SPECIAL	0	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	—	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
STATUS	0	0	0	1	1	1	1	1	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	—	—	—	—	—	—	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	—	—	—	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	—	—	—	0	0	0	0	0	F8h

**DS83C520/DS87C520 SPECIAL FUNCTION REGISTER LOCATIONS** Table 4-7

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	—	—	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	XT/RG	RGMD	RGSL	BGS	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TL_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	—	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TL_1	RI_1	C0h
SBUF1									C1h
ROMSIZE	—	—	—	—	—	RMS2	RMS1	RMS0	C2h
PMR	CD1	CD0	SWB	—	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C7h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	—	—	—	—	—	—	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
ACC									E0h
EIE	—	—	—	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	—	—	—	PWDI	PX5	PX4	PX3	PX2	F8h

Shaded bits are Timed Access protected.

DS83C520/DS87C520 SPECIAL FUNCTION REGISTER RESET VALUES Table 4-8

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	—	—	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	SPECIAL	SPECIAL	SPECIAL	0	91h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	—	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
ROMSIZE	—	—	—	—	—	1	0	1	C2h
PMR	0	1	0	—	0	0	0	0	C4h
STATUS	0	0	0	SPECIAL	0	0	0	0	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	—	—	—	—	—	—	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	—	—	—	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	—	—	—	0	0	0	0	0	F8h



DS87C530 SPECIAL FUNCTION REGISTER LOCATIONS Table 4-9

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	—	—	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	XT/RG	RGMD	RGSL	BGS	91h
TRIM	E4K	X12/6	TRM2	TRM2	TRM1	TRM1	TRM0	TRM0	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	—	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1									C1h
ROMSIZE	—	—	—	—	—	RMS2	RMS1	RMS0	C2h
PMR	CD1	CD0	SWB	—	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C7h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	—	—	—	—	—	—	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
ACC									E0h
EIE	—	—	ERTCI	EWDI	EX5	EX4	EX3	EX2	E8h

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
B									F0h
RTASS									F2h
RTAS	0	0							F3h
RTAM	0	0							F4h
RTAH	0	0	0						F5h
EIP	–	–	PRTCI	PWDI	PX5	PX4	PX3	PX2	F8h
RTCC	SSCE	SCE	MCE	HCE	RTCRE	RTCWE	RTCIF	RTCE	F9h
RTCSS									FAh
RTCS	0	0							FBh
RTCM	0	0							FCh
RTCH									FDh
RTCD0									FEh
RTCD1									FFh

Shaded bits are Timed Access protected.

**DS87C530 SPECIAL FUNCTION REGISTER RESET VALUES** Table 4–10

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP	0	0	0	0	0	1	1	1	81h
DPL	0	0	0	0	0	0	0	0	82h
DPH	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	–	–	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
P1	1	1	1	1	1	1	1	1	90h
EXIF	0	0	0	0	SPECIAL	SPECIAL	SPECIAL	0	91h
TRIM	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	96h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
P2	1	1	1	1	1	1	1	1	A0h
IE	0	0	0	0	0	0	0	0	A8h
SADDR0	0	0	0	0	0	0	0	0	A9h
SADDR1	0	0	0	0	0	0	0	0	AAh
P3	1	1	1	1	1	1	1	1	B0h
IP	–	0	0	0	0	0	0	0	B8h
SADEN0	0	0	0	0	0	0	0	0	B9h
SADEN1	0	0	0	0	0	0	0	0	BAh
SCON1	0	0	0	0	0	0	0	0	C0h

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SBUF1	0	0	0	0	0	0	0	0	C1h
ROMSIZE	—	—	—	—	—	1	0	1	C2h
PMR	0	1	0	—	0	0	0	0	C4h
STATUS	0	0	0	SPECIAL	0	0	0	0	C5h
TA	1	1	1	1	1	1	1	1	C7h
T2CON	0	0	0	0	0	0	0	0	C8h
T2MOD	—	—	—	—	—	—	0	0	C9h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
WDCON	0	SPECIAL	0	SPECIAL	0	SPECIAL	SPECIAL	0	D8h
ACC	0	0	0	0	0	0	0	0	E0h
EIE	—	—	0	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
RTASS	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F2h
RTAS	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F3h
RTAM	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F4h
RTAH	0	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	F5h
EIP	—	—	0	0	0	0	0	0	F8h
RTCC	SPECIAL	SPECIAL	SPECIAL	SPECIAL	0	0	SPECIAL	SPECIAL	F9h
RTCSS	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FAh
RTCS	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FBh
RTCM	0	0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FBh
RTCH	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FDh
RTCD0	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FEh
RTCD1	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	FFh

RC0	0	0	0	0	0	0	0	0	TH0
RC1	0	0	0	0	0	0	0	0	TH1
RC2	1	0	0	0	0	0	0	0	TH2
RC3	1	1	1	1	1	1	1	1	TH3
RC4	0	SPECIAL	SPECIAL	SPECIAL	0	0	0	0	TH4
RC5	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	SPECIAL	TH5
RC6	0	0	0	0	0	0	0	0	TH6
RC7	0	0	0	0	0	0	0	0	TH7
RC8	1	1	1	1	1	1	1	1	TH8
RC9	0	0	0	0	0	0	0	0	TH9
RC10	0	0	0	0	0	0	0	0	TH10
RC11	1	1	1	1	1	1	1	1	TH11
RC12	0	0	0	0	0	0	0	0	TH12
RC13	0	0	0	0	0	0	0	0	TH13
RC14	1	1	1	1	1	1	1	1	TH14
RC15	0	0	0	0	0	0	0	0	TH15
RC16	0	0	0	0	0	0	0	0	TH16
RC17	0	0	0	0	0	0	0	0	TH17
RC18	0	0	0	0	0	0	0	0	TH18
RC19	0	0	0	0	0	0	0	0	TH19
RC20	0	0	0	0	0	0	0	0	TH20
RC21	0	0	0	0	0	0	0	0	TH21
RC22	0	0	0	0	0	0	0	0	TH22
RC23	0	0	0	0	0	0	0	0	TH23
RC24	0	0	0	0	0	0	0	0	TH24
RC25	0	0	0	0	0	0	0	0	TH25
RC26	0	0	0	0	0	0	0	0	TH26
RC27	0	0	0	0	0	0	0	0	TH27
RC28	0	0	0	0	0	0	0	0	TH28
RC29	0	0	0	0	0	0	0	0	TH29
RC30	0	0	0	0	0	0	0	0	TH30
RC31	0	0	0	0	0	0	0	0	TH31

## SPECIAL FUNCTION REGISTERS

Most of the unique features of the High Speed Microcontroller family are controlled by bits in special function registers (SFRs) located in unused locations in the 8051 SFR map. This allows for increased functionality while maintaining complete instruction set compatibility.

The description for each bit indicates its read and write access, as well as its state after a power on reset. Bits which are affected by a no-battery reset are also indicated. Note that many bits and registers are unique to specific devices, and their functions will vary between different members of the High Speed Microcontroller Family.

### Port 0 (P0)

	7	6	5	4	3	2	1	0
SFR 80h	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**P0.7-0**  
Bits 7-0

**Port 0.** This port functions as a multiplexed address/data bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the LSB of the address when ALE is high, and data when ALE is low. When used as a general purpose I/O, this port is open-drain and requires pull-ups. Writing a 1 to any pin of this port will place it in a high impedance mode, which is necessary if the pin is to be used as an input. Pull-ups are not required when used as a memory interface. The Port 0 latch does not exist on devices with no internal program memory, such as the DS80C310 and DS80C320.

### Stack Pointer (SP)

	7	6	5	4	3	2	1	0
SFR 81h	SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-1	RW-1	RW-1

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SP.7-0**  
Bits 7-0

**Stack Pointer.** The stack pointer identifies the location where the stack will begin. The stack pointer is incremented before every PUSH operation. This register defaults to 07h after reset.

### Data Pointer Low 0 (DPL)

	7	6	5	4	3	2	1	0
SFR 82h	DPL.7	DPL.6	DPL.5	DPL.4	DPL.3	DPL.2	DPL.1	DPL.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**DPL.7-0**  
Bits 7-0

**Data Pointer Low 0.** This register is the low byte of the standard 80C32 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.



**Data Pointer High 0 (DPH)**

	7	6	5	4	3	2	1	0
SFR 83h	DPH.7	DPH.6	DPH.5	DPH.4	DPH.3	DPH.2	DPH.1	DPH.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**DPH.7-0**

Bits 7-0

**Data Pointer High 0.** This register is the high byte of the standard 80C32 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.

**Data Pointer Low 1 (DPL1)**

	7	6	5	4	3	2	1	0
SFR 84h	DPL1.7	DPL1.6	DPL1.5	DPL1.4	DPL1.3	DPL1.2	DPL1.1	DPL1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**DPL1.7-0**

Bits 7-0

**Data Pointer Low 1.** This register is the low byte of the auxiliary 16-bit data pointer. When the SEL bit (DPS.0) is set, DPL1 and DPH1 are used in place of DPL and DPH during DPTR operations.

**Data Pointer High 1 (DPH1)**

	7	6	5	4	3	2	1	0
SFR 85h	DPH1.7	DPH1.6	DPH1.5	DPH1.4	DPH1.3	DPH1.2	DPH1.1	DPH1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**DPH1.7-0**

Bits 7-0

**Data Pointer High 1.** This register is the high byte of the auxiliary 16-bit data pointer. When the SEL bit (DPS.0) is set, DPL1 and DPH1 are used in place of DPL and DPH during DPTR operations.

**Data Pointer Select (DPS)**

	7	6	5	4	3	2	1	0
SFR 86h	0	0	0	0	0	0	0	SEL
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

Bits 7-1

Reserved. These bits will read 0.

**SEL**

Bit 0

**Data Pointer Select.** This bit selects the active data pointer.  
 0=Instructions that use the DPTR will use DPL and DPH.  
 1=Instructions that use the DPTR will use DPL1 and DPH1.

**Power Control (PCON)**

	7	6	5	4	3	2	1	0
SFR 87h	SMOD_0	SMOD0	--	--	GF1	GF0	STOP	IDLE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, --n=Value after Reset

**SMOD\_0**

Bit 7

**Serial Port 0 Baud Rate Doubler Enable.** This bit enables/disables the serial baud rate doubling function for Serial Port 0.0=Serial Port 0 baud rate will be that defined by baud rate generation equation.  
1=Serial Port 0 baud rate will be double that defined by baud rate generation equation.**SMOD0**

Bit 6

**Framing Error Detection Enable.** This bit selects function of the SCON0.7 and SCON1.7 bits.

0=SCON0.7 and SCON1.7 control the SM0 function defined for the SCON0 and SCON1 registers.

1=SCON0.7 and SCON1.7 are converted to the Framing Error (FE) flag for the respective Serial Port.

Bits 5-4

Reserved. Read data is indeterminate.

**GF1**

Bit 3

**General Purpose User Flag 1.** This is a bit-addressable, general purpose flag for software control.**GF0**

Bit 2

**General Purpose User Flag 0.** This is a bit-addressable, general purpose flag for software control.**STOP**

Bit 1

**Stop Mode Select.** Setting this bit will stop program execution, halt the CPU oscillator, and internal timers, and place the CPU in a low-power mode. This bit will always be read as a 0.**IDLE**

Bit 0

**Idle Mode Select.** Setting this bit will stop program execution but leave the CPU oscillator, timers, serial ports, and interrupts active. This bit will always be read as a 0.

**Timer/Counter Control (TCON)**

	7	6	5	4	3	2	1	0
SFR 88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

<b>TF1</b> Bit 7	<b>Timer 1 Overflow Flag.</b> This bit indicates when Timer 1 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine. 0=No Timer 1 overflow has been detected. 1=Timer 1 has overflowed its maximum count.
<b>TR1</b> Bit 6	<b>Timer 1 Run Control.</b> This bit enables/disables the operation of Timer 1. Halting this timer will preserve the current count in TH1, TL1. 0=Timer 1 is halted. 1=Timer 1 is enabled.
<b>TF0</b> Bit 5	<b>Timer 0 Overflow Flag.</b> This bit indicates when Timer 0 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine or by software. 0=No Timer 0 overflow has been detected. 1=Timer 0 has overflowed its maximum count.
<b>TR0</b> Bit 4	<b>Timer 0 Run Control.</b> This bit enables/disables the operation of Timer 0. Halting this timer will preserve the current count in TH0, TL0. 0=Timer 0 is halted. 1=Timer 0 is enabled.
<b>IE1</b> Bit 3	<b>Interrupt 1 Edge Detect.</b> This bit is set when an edge/level of the type defined by IT1 is detected. If IT1=1, this bit will remain set until cleared in software or the start of the External Interrupt 1 service routine. If IT1=0, this bit will inversely reflect the state of the $\overline{\text{INT1}}$ pin.
<b>IT1</b> Bit 2	<b>Interrupt 1 Type Select.</b> This bit selects whether the $\overline{\text{INT1}}$ pin will detect edge or level triggered interrupts. 0= $\overline{\text{INT1}}$ is level triggered. 1= $\overline{\text{INT1}}$ is edge triggered.
<b>IE0</b> Bit 1	<b>Interrupt 0 Edge Detect.</b> This bit is set when an edge/level of the type defined by IT0 is detected. If IT0=1, this bit will remain set until cleared in software or the start of the External Interrupt 0 service routine. If IT0=0, this bit will inversely reflect the state of the $\overline{\text{INT0}}$ pin.
<b>IT0</b> Bit 0	<b>Interrupt 0 Type Select.</b> This bit selects whether the $\overline{\text{INT0}}$ pin will detect edge or level triggered interrupts. 0= $\overline{\text{INT0}}$ is level triggered. 1= $\overline{\text{INT0}}$ is edge triggered.

**Timer Mode Control (TMOD)**

	7	6	5	4	3	2	1	0
SFR 89h	GATE	C/T	M1	M0	GATE	C/T	M1	M0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**GATE**  
Bit 7**Timer 1 Gate Control.** This bit enables/disables the ability of Timer 1 to increment.0=Timer 1 will clock when TR1=1, regardless of the state of INT1.  
1=Timer 1 will clock only when TR1=1 and INT1=1.**C/T**  
Bit 6**Timer 1 Counter/Timer Select.**0=Timer is incremented by internal clocks.  
1=Timer is incremented by pulses on T1 when TR1 (TCON.6) is 1.**M1, M0**  
Bits 5-4**Timer 1 Mode Select.** These bits select the operating mode of Timer 1.

M1	M0	Mode
0	0	Mode 0: 8 bits with 5-bit prescale
0	1	Mode 1: 16 bits
1	0	Mode 2: 8 bits with auto-reload
1	1	Mode 3: Timer 1 is halted, but holds its count.

**GATE**  
Bit 3**Timer 0 Gate Control.** This bit enables/disables the ability of Timer 0 to increment.0=Timer 0 will clock when TR0=1, regardless of the state of INT0.  
1=Timer 0 will clock only when TR0=1 and INT0=1.**C/T**  
Bit 2**Timer 0 Counter/Timer Select.**0=Timer is incremented by internal clocks.  
1=Timer is incremented by pulses on T0 when TR0 (TCON.4) is 1.**M1, M0**  
Bits 1-0**Timer 0 Mode Select.** These bits select the operating mode of Timer 0. When timer 0 is in mode 3, TL0 is started/stopped by TR0 and TH0 is started/stopped by TR1. Run control for timer 1 is then provided via the timer 1 mode selection.

M1	M0	Mode
0	0	Mode 0: 8 bits with 5-bit prescale
0	1	Mode 1: 16 bits
1	0	Mode 2: 8 bits with auto-reload
1	1	Mode 3: Timer 0 is two 8 bit counters



**Timer 0 LSB (TL0)**

	7	6	5	4	3	2	1	0
SFR 8Ah	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TL0.7-0** **Timer 0 LSB.** This register contains the least significant byte of Timer 0.  
 Bits 7-0

**Timer 1 LSB (TL1)**

	7	6	5	4	3	2	1	0
SFR 8Bh	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TL1.7-0** **Timer 1 LSB.** This register contains the least significant byte of Timer 1.  
 Bits 7-0

**Timer 0 MSB (TH0)**

	7	6	5	4	3	2	1	0
SFR 8Ch	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TH0.7-0** **Timer 0 MSB.** This register contains the most significant byte of Timer 0.  
 Bits 7-0

**Timer 1 MSB (TH1)**

	7	6	5	4	3	2	1	0
SFR 8Dh	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TH1.7-0** **Timer 1 MSB.** This register contains the most significant byte of Timer 1.  
 Bits 7-0

**Clock Control (CKCON)**

	7	6	5	4	3	2	1	0
SFR 8Eh	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-1

R=Unrestricted Read, W= Unrestricted Write, -n=Value after Reset

**WD1, WD0**

Bits 7-6

**Watchdog Timer Mode Select 1-0.** These bits determine the watchdog timer time-out period. The timer divides the crystal frequency by a programmable value as shown below. The divider value is expressed in clock (crystal) cycles. The use of PMM1 or PMM2 will further divide the clock cycle count by either 16 or 256, respectively. Note that the reset time-out is 512 clocks longer than the interrupt, regardless of whether the interrupt is enabled.

WD1	WD0	Interrupt Divider	Reset Divider
0	0	$2^{17}$	$2^{17} + 512$
0	1	$2^{20}$	$2^{20} + 512$
1	0	$2^{23}$	$2^{23} + 512$
1	1	$2^{26}$	$2^{26} + 512$

**T2M**

Bit 5

**Timer 2 Clock Select.** This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator or clock output modes. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.

0=Timer 2 uses a divide by 12 of the crystal frequency.  
1=Timer 2 uses a divide by 4 of the crystal frequency.

**T1M**

Bit 4

**Timer 1 Clock Select.** This bit controls the division of the system clock that drives Timer 1. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.

0=Timer 1 uses a divide by 12 of the crystal frequency.  
1=Timer 1 uses a divide by 4 of the crystal frequency.

**T0M**

Bit 3

**Timer 0 Clock Select.** This bit controls the division of the system clock that drives Timer 0. Clearing this bit to 0 maintains 80C32 compatibility. This bit has no effect on instruction cycle timing.

0=Timer 0 uses a divide by 12 of the crystal frequency.  
1=Timer 0 uses a divide by 4 of the crystal frequency.

**MD2, MD1, MD0**

Bits 2-0

**Stretch MOVX Select 2-0.** These bits select the time by which external MOVX cycles are to be stretched. This allows slower memory or peripherals to be accessed without using ports or manual software intervention. The RD or WR strobe will be stretched by the specified interval, which will be transparent to the software except for the increased time to execute the MOVX instruction. All internal MOVX instructions on devices containing MOVX SRAM are performed at the 2 machine cycle rate.

MD2	MD1	MD0	Stretch Value	MOVX Duration
0	0	0	0	2 Machine Cycles
0	0	1	1	3 Machine Cycles (reset default)
0	1	0	2	4 Machine Cycles
0	1	1	3	5 Machine Cycles
1	0	0	4	6 Machine Cycles
1	0	1	5	7 Machine Cycles
1	1	0	6	8 Machine Cycles
1	1	1	7	9 Machine Cycles

**Port 1 (P1)**

7	6	5	4	3	2	1	0
P1.7 INT5	P1.6 INT4	P1.5 INT3	P1.4 INT2	P1.3 TXD1	P1.2 RXD1	P1.1 T2EX	P1.0 T2
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**P1.7-0**

Bits 7-0

**General Purpose I/O Port 1.** This register functions as a general purpose I/O port. In addition, all the pins have an alternative function listed below. P1.2-7 contain functions that are new to the 80C32 architecture. The timer 2 functions on pins P1.1-0 are available on the 80C32, but not the 80C31. Each of the functions is controlled by several other SFRs. The associated Port 1 latch bit must contain a logic one before the pin can be used in its alternate function capacity.

**INT5**

Bit 7

**External Interrupt 5.** A falling edge on this pin will cause an external interrupt 5 if enabled.

**INT4**

Bit 6

**External Interrupt 4.** A rising edge on this pin will cause an external interrupt 4 if enabled.

**INT3**

Bit 5

**External Interrupt 3.** A falling edge on this pin will cause an external interrupt 3 if enabled.

**INT2**

Bit 4

**External Interrupt 2.** A rising edge on this pin will cause an external interrupt 2 if enabled.

**TXD1**

Bit 3

**Serial Port 1 Transmit.** This pin transmits the serial port 1 data in serial port modes 1,2,3 and emits the synchronizing clock in serial port mode 0.

**RXD1**

Bit 2

**Serial Port 1 Receive.** This pin receives the serial port 1 data in serial port modes 1,2,3 and is a bi-directional data transfer pin in serial port mode 0.

**T2EX**

Bit 1

**Timer 2 Capture/Reload Trigger.** A 1 to 0 transition on this pin will cause the value in the T2 registers to be transferred into the capture registers if enabled by EXEN2 (T2CON.3). When in auto-reload mode, a 1 to 0 transition on this pin will reload the timer 2 registers with the value in RCAP2L and RCAP2H if enabled by EXEN2 (T2CON.3).

**T2**

Bit 0

**Timer 2 External Input.** A 1 to 0 transition on this pin will cause timer 2 to increment or decrement depending on the timer configuration.

**External Interrupt Flag (EXIF)**

	7	6	5	4	3	2	1	0
SFR 91h	IE5	IE4	IE3	IE2	XT/RG	RGMD	RGSL	BGS
	RW-0	RW-0	RW-0	RW-0	RW-*	R-*	RW-*	RT-0

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, -n=Value after Reset,

\*=See description

**IE5**  
Bit 7

**External Interrupt 5 Flag.** This bit will be set when a falling edge is detected on  $\overline{\text{INT5}}$ . This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

**IE4**  
Bit 6

**External Interrupt 4 Flag.** This bit will be set when a rising edge is detected on  $\text{INT4}$ . This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

**IE3**  
Bit 5

**External Interrupt 3 Flag.** This bit will be set when a falling edge is detected on  $\overline{\text{INT3}}$ . This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

**IE2**  
Bit 4

**External Interrupt 2 Flag.** This bit will be set when a rising edge is detected on  $\text{INT2}$ . This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.

**XT/RG**  
Bit 3

**Crystal/Ring Source Select.** This bit selects the crystal oscillator or ring oscillator as the desired clock source. This bit will be the inverse of RGMD except during the crystal warm-up period when executing a ring oscillator resume from Stop. XTUP (STATUS.4) must be set to 1 and XTOFF (PMR.3) must be cleared to 0 before this bit can be set. Attempts to modify this bit when these conditions are not met will be ignored. This bit must be cleared before XTOFF can be set to 1. This bit is set to 1 after a power-on reset, and unchanged by all other forms of reset. This bit is not used on the DS80C310 or DS80C320 and will be 1 when read.

0=The ring oscillator is selected as the clock source. This setting is unaffected by XTUP (STATUS.4) and XTOFF (PMR.3).

1=The crystal oscillator is selected as the clock source. This setting is invalid unless XTUP=1 and XTOFF=0.

**RGMD**  
Bit 2

**Ring Mode Status.** This bit indicates the current clock source for the device. This bit is cleared to 0 after a power-on reset, and unchanged by all other forms of reset. The state of this bit will be undefined on devices which do not incorporate a ring oscillator.

0=Device is operating from the external crystal or oscillator.

1=Device is operating from the ring oscillator.

**RGSL**  
Bit 1

**Ring Oscillator Select.** This bit selects the clock source following a resume from Stop mode. Using the ring oscillator to resume from Stop mode allows almost instantaneous start-up. This bit is cleared to 0 after a power-on reset, and unchanged by all other forms of reset. The state of this bit will be undefined on devices which do not incorporate a ring oscillator.

0=The device will hold operation until the crystal oscillator has warmed-up.

1=The device will begin operating from the ring oscillator, and when the crys-



tal warm-up is complete, will switch to the clock source indicated by the XT/RG bit.

**BGS**

Bit 0

**Band-gap Select.** This bit enables/disables the band-gap reference during Stop mode. Disabling the band-gap reference provides significant power savings in Stop mode, but sacrifices the ability to perform a power fail interrupt or power-fail reset while stopped. This bit can only be modified with a Timed Access procedure. The state of this bit will be undefined on devices which do not incorporate a band-gap reference.

0=The band-gap reference is disabled in Stop mode but will function during normal operation.

1=The band-gap reference will operate in Stop mode.

**RTC Trim Register (TRIM)**

	7	6	5	4	3	2	1	0
SFR 96h	E4K	X12/6	TRM2	TRM2	TRM1	TRM1	TRM0	TRM0
	RT-*	RT-*	RT-*	RT-*	RT-*	RT-*	RT-*	RT-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

**E4K**

Bit 7

**External 4096 Hz RTC Signal Enable.** This bit enables the output of a 4096 Hz signal on pin P1.7 derived from the RTC. Setting this bit overrides any other function of the pin. It is used for adjusting the frequency of the 32.768 RTC crystal oscillator using the trim bits. This bit is cleared to 0 after any reset, including a no-battery reset.

0=Calibration function disabled. P1.7 pin will function per the normal pin description.

1=4096 Hz signal output on P1.7

**X12/6**

Bit 6

**RTC Crystal Capacitance Select.** This bit selects the internal loading capacitance of the RTC crystal amplifier. This bit is set to 1 after a no-battery reset, and unchanged by all other forms of reset.

0=RTC loading is set for 6 pF crystal.

1=RTC loading is set for 12.5 pF crystal.

**TRM2**

Bit 5

**RTC Trim Bit 2.** This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 1 after a no-battery reset, and unchanged by all other forms of reset.

**TRM2**

Bit 4

**RTC Inverted Trim Bit 2.** This bit must always be set to the complement of the TRM2 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5-0 to their no-battery reset value. This bit is cleared to 0 after a no-battery reset, and unchanged by all other forms of reset.

**TRM1**

Bit 3

**RTC Trim Bit 1.** This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 0 after a no-battery reset, and unchanged by all other forms of reset.

**TRM1**

Bit 2

**RTC Inverted Trim Bit 1.** This bit must always be set to the complement of the TRM1 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5-0 to their no-battery reset value. This bit is cleared to 1 after a no-battery reset, and unchanged by all other forms of reset.

**TRM0**

Bit 1

**RTC Trim Bit 0.** This bit controls the relative adjustment of the RTC internal capacitance. It is used to calibrate the RTC oscillator frequency. This bit is set to 0 after a no-battery reset, and unchanged by all other forms of reset.

**TRM0**

Bit 0

**RTC Inverted Trim Bit 0.** This bit must always be set to the complement of the TRM0 bit. Incorrectly writing this bit will default bits TRIM.7, TRIM.5–0 to their no-battery reset value. This bit is cleared to 1 after a no-battery reset, and unchanged by all other forms of reset.

**Serial Port 0 Control (SCON0)**

	7	6	5	4	3	2	1	0
SFR 98h	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

**SM0–2**

Bits 7–5

**Serial Port 0 Mode.** These bits control the mode of serial port 0. In addition, the SM0 and SM2\_0 bits have secondary functions as shown below.

SM0	SM1	SM2	MODE	FUNCTION	LENGTH	PERIOD
0	0	0	0	Synchronous	8 bits	12 $t_{CLK}$
0	0	1	0	Synchronous	8 bits	4 $t_{CLK}$
0	1	x	1	Asynchronous	10 bits	Timer 1 or 2 baud rate equation
1	0	0	2	Asynchronous	11 bits	64 $t_{CLK}$ (SMOD=0) 32 $t_{CLK}$ (SMOD=1)
1	0	1	2	Asynchronous w/ Multiprocessor communication	11 bits	64 $t_{CLK}$ (SMOD=0) 32 $t_{CLK}$ (SMOD=1)
1	1	0	3	Asynchronous	11 bits	Timer 1 or 2 baud rate equation
1	1	1	3	Asynchronous w/ Multiprocessor communication	11 bits	Timer 1 or 2 baud rate equation

**SM0/FE\_0**

Bit 7

**Framing Error Flag.** When SMOD0 (PCON.6)=0, this bit (SM0) is used to select the mode for serial port 0. When SMOD0 (PCON.6)=1, this bit (FE) will be set upon detection of an invalid stop bit. When used as FE, this bit must be cleared in software. Once the SMOD0 bit is set, modifications to this bit will not affect the serial port mode settings. Although accessed from the same register, internally the data for bits SM0 and FE are stored in different locations.

**SM1\_0**

Bit 6

**No alternate function.**

**SM2\_0**

Bit 5

**Multiple CPU Communications.** The function of this bit is dependent on the serial port 0 mode.

Mode 0: Selects 12  $t_{CLK}$  or 4  $t_{CLK}$  period for synchronous serial port 0 data transfers.

Mode 1: When set, reception is ignored (RI\_0 is not set) if invalid stop bit received.

Mode 2/3: When this bit is set, multiprocessor communications are enabled

in modes 2 and 3. This will prevent the RI\_0 bit from being set, and an interrupt being asserted, if the 9th bit received is not 1.

**REN\_0**

Bit 4

**Receive Enable.** This bit enables/disables the serial port 0 receiver shift register.

0=Serial port 0 reception disabled.

1=Serial port 0 receiver enabled (modes 1, 2, 3). Initiate synchronous reception (mode 0).

**TB8\_0**

Bit 3

**9th Transmission Bit State.** This bit defines the state of the 9th transmission bit in serial port 0 modes 2 and 3.

**RB8\_0**

Bit 2

**9th Received Bit State.** This bit identifies the state of the 9th reception bit of received data in serial port 0 modes 2 and 3. In serial port mode 1, when SM2\_0=0, RB8\_0 is the state of the stop bit. RB8\_0 is not used in mode 0.

**TI\_0**

Bit 1

**Transmitter Interrupt Flag.** This bit indicates that data in the serial port 0 buffer has been completely shifted out. In serial port mode 0, TI\_0 is set at the end of the 8th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.

**RI\_0**

Bit 0

**Receiver Interrupt Flag.** This bit indicates that a byte of data has been received in the serial port 0 buffer. In serial port mode 0, RI\_0 is set at the end of the 8th bit. In serial port mode 1, RI\_0 is set after the last sample of the incoming stop bit subject to the state of SM2\_0. In modes 2 and 3, RI\_0 is set after the last sample of RB8\_0. This bit must be manually cleared by software.

**Serial Data Buffer 0 (SBUF0)**

	7	6	5	4	3	2	1	0
SFR 99h	SBUF0.7	SBUF0.6	SBUF0.5	SBUF0.4	SBUF0.3	SBUF0.2	SBUF0.1	SBUF0.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SBUF0.7-0**

Bits 7-0

**Serial Data Buffer 0.** Data for serial port 0 is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at this location.

**Port 2 (P2)**

	7	6	5	4	3	2	1	0
SFR A0h	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**P2.7-0**  
Bits 7-0

**Port 2.** This port functions as an address bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the MSB of the address. The Port 2 latch does not control general purpose I/O pins on the DS80C310 and DS80C320, but is still used to hold the address MSB during register-indirect data memory operations such as MOVX A, @R1.

**Interrupt Enable (IE)**

	7	6	5	4	3	2	1	0
SFR A8h	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**EA**  
Bit 7

**Global Interrupt Enable.** This bit controls the global masking of all interrupts except Power-Fail Interrupt, which is enabled by the EPFI bit (WDCON.5).  
0=Disable all interrupt sources. This bit overrides individual interrupt mask settings.  
1=Enable all individual interrupt masks. Individual interrupts will occur if enabled.

**ES1**  
Bit 6

**Enable Serial Port 1 Interrupt.** This bit controls the masking of the serial port 1 interrupt.  
0=Disable all serial port 1 interrupts.  
1=Enable interrupt requests generated by the RI\_1 (SCON1.0) or TI\_1 (SCON1.1) flags.

**ET2**  
Bit 5

**Enable Timer 2 Interrupt.** This bit controls the masking of the Timer 2 interrupt.  
0=Disable all Timer 2 interrupts.  
1=Enable interrupt requests generated by the TF2 flag (T2CON.7).

**ES0**  
Bit 4

**Enable Serial port 0 Interrupt.** This bit controls the masking of the serial port 0 interrupt.  
0=Disable all serial port 0 interrupts.  
1=Enable interrupt requests generated by the RI\_0 (SCON0.0) or TI\_0 (SCON0.1) flags.



**ET1**  
Bit 3

**Enable Timer 1 Interrupt.** This bit controls the masking of the Timer 1 interrupt.

0=Disable all Timer 1 interrupts.

1=Enable interrupt requests generated by the TF1 flag (TCON.7).

**EX1**  
Bit 2

**Enable External Interrupt 1.** This bit controls the masking of external interrupt 1.

0=Disable external interrupt 1.

1=Enable interrupt requests generated by the  $\overline{\text{INT1}}$  pin.

**ET0**  
Bit 1

**Enable Timer 0 Interrupt.** This bit controls the masking of the Timer 0 interrupt.

0=Disable all Timer 0 interrupts.

1=Enable interrupt requests generated by the TF0 flag (TCON.5).

**EX0**  
Bit 0

**Enable External Interrupt 0.** This bit controls the masking of external interrupt 0.

0=Disable external interrupt 0.

1=Enable interrupt requests generated by the  $\overline{\text{INT0}}$  pin.

### Slave Address Register 0 (SADDR0)

	7	6	5	4	3	2	1	0
SFR A9h	SADDR0.7	SADDR0.6	SADDR0.5	SADDR0.4	SADDR0.3	SADDR0.2	SADDR0.1	SADDR0.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SADDR0.7-0**  
Bits 7-0

**Slave Address Register 0.** This register is programmed with the given or broadcast address assigned to serial port 0.

### Slave Address Register 1 (SADDR1)

	7	6	5	4	3	2	1	0
SFR AAh	SADDR1.7	SADDR1.6	SADDR1.5	SADDR1.4	SADDR1.3	SADDR1.2	SADDR1.1	SADDR1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SADDR1.7-0**  
Bits 7-0

**Slave Address Register 1.** This register is programmed with the given or broadcast address assigned to serial port 1.

**Port 3 (P3)**

	7	6	5	4	3	2	1	0
SFR B0h	P3.7 RD	P3.6 WR	P3.5 T1	P3.4 T0	P3.3 INT1	P3.2 INT0	P3.1 TXD0	P3.0 RXD0
	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**P3.7-0**  
Bits 7-0

**General Purpose I/O Port 3.** This register functions as a general purpose I/O port. In addition, all the pins have an alternative function listed below. Each of the functions is controlled by several other SFRs. The associated Port 1 latch bit must contain a logic one before the pin can be used in its alternate function capacity.

**RD**  
Bit 7

**External Data Memory Read Strobe.** This pin provides an active low read strobe to an external memory device.

**WR**  
Bit 6

**External Data Memory Write Strobe.** This pin provides an active low write strobe to an external memory device.

**T1**  
Bit 5

**Timer/Counter 1 External Input.** A 1 to 0 transition on this pin will increment timer 1.

**T0**  
Bit 4

**Timer/Counter 0 External Input.** A 1 to 0 transition on this pin will increment timer 0.

**INT1**  
Bit 3

**External Interrupt 1.** A falling edge/ low level on this pin will cause an external interrupt 1 if enabled.

**INT0**  
Bit 2

**External Interrupt 0.** A falling edge/ low level on this pin will cause an external interrupt 0 if enabled.

**TXD0**  
Bit 1

**Serial Port 0 Transmit.** This pin transmits the serial port 0 data in serial port modes 1,2,3 and emits the synchronizing clock in serial port mode 0.

**RXD0**  
Bit 0

**Serial Port 0 Receive.** This pin receives the serial port 0 data in serial port modes 1,2,3 and is a bi-directional data transfer pin in serial port mode 0.

**Interrupt Priority (IP)**

	7	6	5	4	3	2	1	0
SFR B8h	—	PS1	PT2	PS0	PT1	PX1	PT0	PX0
		RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

Bit 7

Reserved. Read data is indeterminate.

**PS1**  
Bit 6

**Serial Port 1 Interrupt.** This bit controls the priority of the serial port 1 interrupt.  
0=Serial port 1 priority is determined by the natural priority order.  
1=Serial port 1 is a high priority interrupt.

**PT2**

Bit 5

**Timer 2 Interrupt.** This bit controls the priority of the Timer 2 interrupt.

0=Timer 2 priority is determined by the natural priority order.

1=Timer 2 is a high priority interrupt.

**PS0**

Bit 4

**Serial port 0 Interrupt.** This bit controls the priority of the serial port 0 interrupt.

0=Serial port 0 priority is determined by the natural priority order.

1=Serial port 0 is a high priority interrupt.

**PT1**

Bit 3

**Timer 1 Interrupt.** This bit controls the priority of the Timer 1 interrupt.

0=Timer 1 priority is determined by the natural priority order.

1=Timer 1 is a high priority interrupt.

**PX1**

Bit 2

**External Interrupt 1.** This bit controls the priority of external interrupt 1.

0=External interrupt 1 priority is determined by the natural priority order.

1=External interrupt 1 is a high priority interrupt.

**PT0**

Bit 1

**Timer 0 Interrupt.** This bit controls the priority of the Timer 0 interrupt.

0=Timer 0 priority is determined by the natural priority order.

1=Timer 0 is a high priority interrupt.

**PX0**

Bit 0

**External Interrupt 0.** This bit controls the priority of external interrupt 0.

0=External interrupt 0 priority is determined by the natural priority order.

1=External interrupt 0 is a high priority interrupt.

**Slave Address Mask Enable Register 0 (SADEN0)**

	7	6	5	4	3	2	1	0
SFR B9h	SADEN0.7	SADEN0.6	SADEN0.5	SADEN0.4	SADEN0.3	SADEN0.2	SADEN0.1	SADEN0.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SADEN0.7-0**

Bits 7-0

**Slave Address Mask Enable Register 0.** This register functions as a mask when comparing serial port 0 addresses for automatic address recognition. When a bit in this register is set, the corresponding bit location in the SADDR0 register will be exactly compared with the incoming serial port 0 data to determine if a receiver interrupt should be generated. When a bit in this register is cleared, the corresponding bit in the SADDR0 register becomes a don't care and is not compared against the incoming data. All incoming data will generate a receiver interrupt when this register is cleared.

**Slave Address Mask Enable Register 1 (SADEN1)**

	7	6	5	4	3	2	1	0
SFR BAh	SADEN1.7	SADEN1.6	SADEN1.5	SADEN1.4	SADEN1.3	SADEN1.2	SADEN1.1	SADEN1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SADEN1.7-0**

Bits 7-0

**Slave Address Mask Enable Register 1.** This register functions as a mask when comparing serial port 1 addresses for automatic address recognition.

When a bit in this register is set, the corresponding bit location in the SADDR1 register will be exactly compared with the incoming serial port 1 data to determine if a receiver interrupt should be generated. When a bit in this register is cleared, the corresponding bit in the SADDR1 register becomes a don't care and is not compared against the incoming data. All incoming data will generate a receiver interrupt when this register is cleared.

### Serial Port 1 Control (SCON1)

	7	6	5	4	3	2	1	0
SFR C0h	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SM0-2,**  
Bits 7-5

**Serial Port 1 Mode.** These bits control the mode of serial port 1 as shown below. In addition, the SM0 and SM2 bits have secondary functions as shown below.

SM0	SM1	SM2	MODE	FUNCTION	LENGTH	PERIOD
0	0	0	0	Synchronous	8 bits	12 $t_{CLK}$
0	0	1	0	Synchronous	8 bits	4 $t_{CLK}$
0	1	x	1	Asynchronous	10 bits	Timer 1 baud rate equation
1	0	0	2	Asynchronous	11 bits	64 $t_{CLK}$ (SMOD=0) 32 $t_{CLK}$ (SMOD=1)
1	0	1	2	Asynchronous w/ Multiprocessor communication	11 bits	64 $t_{CLK}$ (SMOD=0) 32 $t_{CLK}$ (SMOD=1)
1	1	0	3	Asynchronous	11 bits	Timer 1 baud rate equation
1	1	1	3	Asynchronous w/ Multiprocessor communication	11 bits	Timer 1 baud rate equation

**SM0/FE\_1**  
Bit 7

**Framing Error Flag.** When SMOD0 (PCON.6)=0, this bit (SM0) is used to select the mode for serial port 1. When SMOD0 (PCON.6)=1, this bit (FE) will be set upon detection of an invalid stop bit. When used as FE, this bit must be cleared in software. Once the SMOD0 bit is set, modifications to this bit will not affect the serial port mode settings. Although accessed from the same register, internally the data for bits SM0 and FE are stored in different locations.

**SM1\_1**  
Bit 6

**No alternate function.**

**SM2\_1**  
Bit 5

**Multiple CPU Communications.** The function of this bit is dependent on the serial port 1 mode.

Mode 0: Selects 12  $t_{CLK}$  or 4  $t_{CLK}$  period for synchronous serial port 1 data transfers.

Mode 1: When this bit is set, reception is ignored (RI\_1 is not set) if invalid stop bit received.

Mode 2/3: When this bit is set, multiprocessor communications are enabled



in modes 2 and 3. This will prevent the RI\_1 bit from being set, and an interrupt being asserted, if the 9th bit received is not 1.

**REN\_1**  
Bit 4

**Receive Enable.** This bit enables/disables the serial port 1 receiver shift register.

0=Serial port 1 reception disabled.

1=Serial port 1 receiver enabled (modes 1, 2, 3). Initiate synchronous reception (mode 0).

**TB8\_1**  
Bit 3

**9th Transmission Bit State.** This bit defines the state of the 9th transmission bit in serial port 1 modes 2 and 3.

**RB8\_1**  
Bit 2

**9th Received Bit State.** This bit identifies the state of the 9th reception bit of received data in serial port 1 modes 2 and 3. In serial port mode 1, when SM2\_1=0, RB8\_1 is the state of the stop bit. RB8\_1 is not used in mode 0.

**TI\_1**  
Bit 1

**Transmitter Interrupt Flag.** This bit indicates that data in the serial port 1 buffer has been completely shifted out. In serial port mode 0, TI\_1 is set at the end of the 8th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.

**RI\_1**  
Bit 0

**Receiver Interrupt Flag.** This bit indicates that a byte of data has been received in the serial port 1 buffer. In serial port mode 1, RI\_1 is set at the end of the 8th bit. In serial port mode 1, RI\_1 is set after the last sample of the incoming stop bit subject to the state of SM2\_1. In modes 2 and 3, RI\_1 is set after the last sample of RB8\_1. This bit must be manually cleared by software.

#### Serial Data Buffer 1 (SBUF1)

	7	6	5	4	3	2	1	0
SFR C1h	SBUF1.7	SBUF1.6	SBUF1.5	SBUF1.4	SBUF1.3	SBUF1.2	SBUF1.1	SBUF1.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**SBUF1.7-0**  
Bits 7-0

**Serial Data Buffer 1.** Data for serial port 1 is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at this location.

#### ROM Size Select (ROMSIZE)

	7	6	5	4	3	2	1	0
SFR C2h	-	-	-	-	-	RS2	RS1	RS0
						RT-1	RT-0	RT-1

R=Unrestricted Read, T=Timed Access Write Only, -n=Value after Reset

Bits 7-3

These bits are reserved. Read data is indeterminate.

**ROMSIZE.2-0**  
Bits 2-0

**ROM Size Select 2-0.** This register is used to select the maximum on-chip decoded address for ROM. Care must be taken that the memory location of the current program counter will be valid both before and after modification.

These bits can only be modified using a timed access procedure. The  $\overline{EA}$  pin will override the function of these bits when asserted, forcing the device to access external program memory only. Configuring this register to a setting that exceeds the maximum amount of internal memory may corrupt device operation. These bits will default on reset to the maximum amount of internal program memory (i.e. 16K for DS87C520).

RS2	RS1	RS0	MAXIMUM ON-CHIP ROM ADDRESS
0	0	0	0KB/Disable on-chip ROM
0	0	1	1KB/03FFh
0	1	0	2KB/07FFh
0	1	1	4KB/0FFFh
1	0	0	8KB/1FFFh
1	0	1	16KB/3FFFh
1	1	0	32KB/7FFFh
1	1	1	64KB/FFFFh

### Power Management Register (PMR)

7	6	5	4	3	2	1	0
CD1	CD0	SWB	–	XTOFF	ALEOFF	DME1	DME0
RW–0	RW–1	RW–0		RW*–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, \*=See description

**CD1, CD0**  
Bits 7–6

**Clock Divide Control 1–0.** These bits select the number of crystal oscillator clocks required to generate one machine cycle. Switching between modes requires a transition through the divide by 4 mode (CD1, CD0=01). For example, to go from 64 to 1024 clocks per cycle the device must first go from 64 to 4 clocks per cycle, and then from 4 to 1024 clocks per cycle. Attempts to perform an invalid transition will be ignored. The setting of these bits will effect the timers and serial ports as shown below.

		OSC CYCLES PER MACH. CYCLE	OSC CYCLES PER TIMER 0/1/2 CLOCK		OSC CYCLES PER TIMER 2 CLK, BAUD RATE GEN.		OSC CYCLES PER SERIAL PORT CLK, MODE 0		OSC CYCLES PER SERIAL PORT CLK, MODE 2	
CD1	CD0		TxM=0	TxM=1	T2M=0	T2M=1	SM2=0	SM2=1	SDMO=0	SMOD=1
0	0		RESERVED							
0	1	4	12	4	2	2	12	4	64	32
1	0	64	192	64	32	32	192	64	1024	512
1	1	1024	3072	1024	512	512	3072	1024	16384	8192

**SWB**  
Bit 5

**Switchback Enable.** This bit allows an enabled external interrupt or serial port activity to force the Clock Divide Control bits to the divide by 4 state (01). Upon internal acknowledgment of an external interrupt, the device will

switch modes at the start of the jump to the interrupt service routine. Note that this means that an external interrupt must actually be recognized (i.e. be enabled and not masked by higher priority interrupts) for the switchback to occur. For serial port reception, the switch occurs at the start of the instruction following the falling edge of the start bit.

Bit 4 Reserved. When modifying the PMR register, software must write a 0 to this bit. Read data will be indeterminate.

#### XTOFF

Bit 3

**Crystal Oscillator Disable.** This bit disables the CPU crystal oscillator. It can only be set to 1 while running from the ring oscillator ( $XT/\overline{RG}=0$ ). Clearing this bit restarts the crystal amplifier, resets the crystal warm-up counter, and after 65,536 external crystal cycles the XTUP bit will be set.

0=Crystal oscillator is enabled

1=Crystal oscillator is disabled

#### ALEOFF

Bit 2

**ALE Disable.** This bit disables the expression of the ALE signal on the device pin during all on-board program and data memory accesses. External memory accesses will automatically enable ALE independent of ALE-OFF.

0=ALE expression is enabled

1=ALE expression is disabled

#### DME1, DME0

Bits 1–0

**Data Memory Enable 1–0.** These bits determine the functional relationship of the first 1024 bytes of data memory. Three memory configurations are supported to allow either external data memory access through the expanded multiplexed address/data bus of Ports 0 and Port 2, internal SRAM data memory access, or read-only access to EPROM programming information. Note these bits are cleared after a reset, so access to the internal SRAM is prohibited until these bits are modified.

DME1	DME0	DATA MEMORY ADDRESS RANGE	MEMORY ACCESS
0	0	0000h – FFFFh	External Data Memory (default)
0	1	0000h – 03FFh 0400h – FFFFh	Internal SRAM Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h – 03FFh 0400h – FFFBh FFFC FFFDh – FFFFh	Internal SRAM Data Memory Reserved System Control Byte (EPROM Read Only) Reserved

The System Control Byte is a special EPROM location that contains nonvolatile system information. This byte is set during EPROM programming and is not alterable by software. This register can only be read when both Data Memory Enable bits are set. The user must be sure that this location is programmed by the use of a special programming utility supplied with the programming device.

**System Control Byte Description (EPROM; FFFCh)**

Bits 7–3

Reserved. These bits will read 1. These bits should be set to 1 during EPROM programming.

**LB3, LB2, LB1**

Bits 2–0

**EPROM Program Lock Bit 3–1.** These bits show the status of the firmware security of the on-board EPROM. Bit combinations other than shown are illegal.

LB3	LB2	LB1	EPROM PROTECTION MODE
0	0	0	Unconditional verification, full external operation. Additional EPROM programming allowed without full device erasure.
0	0	1	Verification using encryption, execution of external MOVC instruction on internal program memory is disabled. All other program execution and data memory access allowed. Device must be fully erased before EPROM can be programmed again.
0	1	1	Verification disabled, execution of external MOVC instruction on internal program memory is disabled, and access to internal MOVX data from external program is prohibited. All other program execution and data memory access allowed. Device must be fully erased before EPROM can be programmed again.
1	1	1	Verification disabled, external program execution prohibited. Device must be fully erased before EPROM can be programmed again.

**Status Register (STATUS)**

	7	6	5	4	3	2	1	0
SFR C5h	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0
	R–0	R–0	R–0	R–*	R–0	R–0	R–0	R–0

R=Unrestricted Read, –n=Value after Reset, \*=See description

**PIP**

Bit 7

**Power Fail Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a power–fail interrupt. It is cleared when the program executes the corresponding RETI instruction. This bit is indeterminate on devices which do not incorporate the power–fail interrupt.

**HIP**

Bit 6

**High Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a high priority interrupt. It is cleared when the program executes the corresponding RETI instruction.

**LIP**

Bit 5

**Low Priority Interrupt Status.** When set, this bit indicates that software is currently servicing a low priority interrupt. It is cleared when the program executes the corresponding RETI instruction.

**XTUP**

Bit 4

**Crystal Oscillator Warm–up Status.** This bit indicates whether the CPU crystal oscillator has completed the 65,536 cycle warm–up and is ready to operate from the external crystal or oscillator. This bit is cleared each time the crystal oscillator is restarted following an exit from Stop mode or the XTOFF bit (PMR.3) is set. While cleared, this bit prevents software from setting the XT/RG bit (EXIF.3) to enable operation from the crystal. Note that XTUP differs from the RGMD bit (EXIF.2) in that XTUP shows the status of



the crystal while RGMD shows the current clock source. This bit is set to 1 following a power-on reset, but is unaffected by other forms of reset.

**SPTA1**  
Bit 3

**Serial Port 1 Transmit Activity Monitor.** When set, this bit indicates that data is currently being transmitted by serial port 1. It is cleared when the internal hardware sets the TI\_1 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

**SPRA1**  
Bit 2

**Serial Port 1 Receive Activity Monitor.** When set, this bit indicates that data is currently being received by serial port 1. It is cleared when the internal hardware sets the RI\_1 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

**SPTA0**  
Bit 1

**Serial Port 0 Transmit Activity Monitor.** When set, this bit indicates that data is currently being transmitted by serial port 0. It is cleared when the internal hardware sets the TI\_0 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

**SPRA0**  
Bit 0

**Serial Port 0 Receive Activity Monitor.** When set, this bit indicates that data is currently being received by serial port 0. It is cleared when the internal hardware sets the RI\_0 bit. Do not alter the Clock Divide Control bits (PMR.7–6) while this bit is set or serial port data may be lost.

#### Timed Access Register (TA)

	7	6	5	4	3	2	1	0
SFR C7h	TA.7	TA.6	TA.5	TA.4	TA.3	TA.2	TA.1	TA.0
	W-1	W-1	W-1	W-1	W-1	W-1	W-1	W-1

W=Unrestricted Write, -n=Value after Reset

**TA.7–0**  
Bits 7–0

**Timed Access.** Correctly accessing this register permits modification of timed access protected bits. Write AAh to this register first, followed within 3 cycles by writing 55h. Timed access protected bits can then be modified for a period of 3 cycles measured from the writing of the 55h.

#### Timer 2 Control (T2CON)

	7	6	5	4	3	2	1	0
SFR C8h	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TF2**  
Bit 7

**Timer 2 Overflow Flag.** This flag will be set when Timer 2 overflows from FFFFh or the count equal to the capture register in down count mode. It must be cleared by software. TF2 will only be set if RCLK and TCLK are both cleared to 0.

**EXF2**  
Bit 6

**Timer 2 External Flag.** A negative transition on the T2EX pin (P1.1) or timer 2 underflow/overflow will cause this flag to set based on the CP/RL2 (T2CON.0), EXEN2 (T2CON.3), and DCEN (T2MOD.0) bits. If set by a neg-

ative transition, this flag must be cleared to 0 by software. Setting this bit in software or detection of a negative transition on the T2EX pin will force a timer interrupt if enabled.

CP/RL2	EXEN2	DCEN	RESULT
1	0	x	Negative transitions on P1.1 will not affect this bit.
1	1	x	Negative transitions on P1.1 will set this bit.
0	0	0	Negative transitions on P1.1 will not affect this bit.
0	1	0	Negative transitions on P1.1 will set this bit.
0	x	1	Bit toggles whenever timer 2 underflows/overflows and can be used as a 17th bit of resolution. In this mode, EXF2 will not cause an interrupt.

**RCLK**  
Bit 5

**Receive Clock Flag.** This bit determines the serial port 0 timebase when receiving data in serial modes 1 or 3.

0=Timer 1 overflow is used to determine receiver baud rate for serial port 0.  
1=Timer 2 overflow is used to determine receiver baud rate for serial port 0. Setting this bit will force timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.

**TCLK**  
Bit 4

**Transmit Clock Flag.** This bit determines the serial port 0 timebase when transmitting data in serial modes 1 or 3.

0=Timer 1 overflow is used to determine transmitter baud rate for serial port 0.  
1=Timer 2 overflow is used to determine transmitter baud rate for serial port 0. Setting this bit will force timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.

**EXEN2**  
Bit 3

**Timer 2 External Enable.** This bit enables the capture/ reload function on the T2EX pin if Timer 2 is not generating baud rates for the serial port.

0=Timer 2 will ignore all external events at T2EX.  
1=Timer 2 will capture or reload a value if a negative transition is detected on the T2EX pin.

**TR2**  
Bit 2

**Timer 1 Run Control.** This bit enables/disables the operation of timer 2.

Halting this timer will preserve the current count in TH2, TL2.

0=Timer 2 is halted.  
1=Timer 2 is enabled.

**C/T2**  
Bit 1

**Counter/Timer Select.** This bit determines whether timer 2 will function as a timer or counter. Independent of this bit, timer 2 runs at 2 clocks per tick when used in either baud rate generator or clock output mode.

0=Timer 2 function as a timer. The speed of timer 2 is determined by the T2M bit (CKCON.5).  
1=Timer 2 will count negative transitions on the T2 pin (P1.0).

**CP/RL2**  
Bit 0

**Capture/Reload Select.** This bit determines whether the capture or reload function will be used for timer 2. If either RCLK or TCLK is set, this bit will not function and the timer will function in an auto-reload mode following each overflow.

0=Auto-reloads will occur when timer 2 overflows or a falling edge is detected on T2EX if EXEN2=1.

1=Timer 2 captures will occur when a falling edge is detected on T2EX if EXEN2=1.

### Timer 2 Mode (T2MOD)

	7	6	5	4	3	2	1	0
SFR C9h	—	—	—	—	—	—	T2OE	DCEN
							RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, —n=Value after Reset

Bits 7–2

Reserved. Read data will be indeterminate.

**T2OE**

Bit 1

**Timer 2 Output Enable.** This bit enables/disables the clock output function of the T2 pin (P1.0).

0=The T2 pin functions as either a standard port pin or as a counter input for timer 2.

1=Timer 2 will drive the T2 pin with a clock output if  $C/\overline{T2}=0$ . Also, timer 2 rollovers will not cause interrupts.

**DCEN**

Bit 0

**Down Count Enable.** This bit, in conjunction with the T2EX pin, controls the direction that timer 2 counts in 16-bit auto-reload mode.

DCEN	T2EX	DIRECTION
1	1	Up
1	0	Down
0	x	Up

### Timer 2 Capture LSB (RCAP2L)

	7	6	5	4	3	2	1	0
SFR CAh	RCAP2L.7	RCAP2L.6	RCAP2L.5	RCAP2L.4	RCAP2L.3	RCAP2L.2	RCAP2L.1	RCAP2L.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, —n=Value after Reset

**RCAP2L.7–0**

Bits 7–0

**Timer 2 Capture LSB.** This register is used to capture the TL2 value when timer 2 is configured in capture mode. RCAP2L is also used as the LSB of a 16-bit reload value when timer 2 is configured in auto-reload mode.

**Timer 2 Capture MSB (RCAP2H)**

	7	6	5	4	3	2	1	0
SFR CBh	RCAP2H.7	RCAP2H.6	RCAP2H.5	RCAP2H.4	RCAP2H.3	RCAP2H.2	RCAP2H.1	RCAP2H.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**RCAP2H.7-0**

Bits 7-0

**Timer 2 Capture MSB.** This register is used to capture the TH2 value when timer 2 is configured in capture mode. RCAP2H is also used as the MSB of a 16-bit reload value when timer 2 is configured in auto-reload mode.

**Timer 2 LSB (TL2)**

	7	6	5	4	3	2	1	0
SFR CCh	TL2.7	TL2.6	TL2.5	TL2.4	TL2.3	TL2.2	TL2.1	TL2.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TL2.7-0**

Bits 7-0

**Timer 2 LSB.** This register contains the least significant byte of Timer 2.

**Timer 2 MSB (TH2)**

	7	6	5	4	3	2	1	0
SFR CDh	TH2.7	TH2.6	TH2.5	TH2.4	TH2.3	TH2.2	TH2.1	TH2.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**TH2.7-0**

Bits 7-0

**Timer 2 MSB.** This register contains the most significant byte of Timer 2.

**Program Status Word (PSW)**

	7	6	5	4	3	2	1	0
SFR D0h	CY	AC	F0	RS1	RS0	OV	F1	PARITY
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**CY**

Bit 7

**Carry Flag.** This bit is set when if the last arithmetic operation resulted in a carry (during addition) or a borrow (during subtraction). Otherwise it is cleared to 0 by all arithmetic operations.

**AC**

Bit 6

**Auxiliary Carry Flag.** This bit is set to 1 if the last arithmetic operation resulted in a carry into (during addition), or a borrow (during subtraction) from the high order nibble. Otherwise it is cleared to 0 by all arithmetic operations.

**F0**

Bit 5

**User Flag 0.** This is a bit-addressable, general purpose flag for software control.

**RS1, RS0**

Bits 4–3

**Register Bank Select 1–0.** These bits select which register bank is addressed during register accesses.

RS1	RS0	REGISTER BANK	ADDRESS
0	0	0	00h – 07h
0	1	1	08h – 0Fh
1	0	2	10h – 17h
1	1	3	18h – 1Fh

**OV**

Bit 2

**Overflow Flag.** This bit is set to 1 if the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise it is cleared to 0 by all arithmetic operations.

**F1**

Bit 1

**User Flag 1.** This is a bit-addressable, general purpose flag for software control.

**PARITY**

Bit 0

**Parity Flag.** This bit is set to 1 if the modulo-2 sum of the eight bits of the accumulator is 1 (odd parity); and cleared to 0 on even parity.

**Watchdog Control (WDCON)**

	7	6	5	4	3	2	1	0
SFR D8h	SMOD	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT
	RW-0	RT-*	RW-0	RW-*	RT-0	RW-*	RT-*	RT-0

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, -n=Value after Reset,

\*=See description

**SMOD**

Bit 7

**Serial Modification.** This bit controls the doubling of the serial port 1 baud rate in modes 1, 2, and 3.

0=Serial port 1 baud rate operates at normal speed

1=Serial port 1 baud rate is doubled.

**POR**

Bit 6

**Power-on Reset Flag.** This bit indicates whether the last reset was a power-on reset. This bit is typically interrogated following a reset to determine if the reset was caused by a power-on reset. It must be cleared by a Timed Access write before the next reset of any kind or the software may erroneously determine that another power-on reset has occurred. This bit is set following a power-on reset and unaffected by all other resets.

0=Last reset was from a source other than a power-on reset

1=Last reset was a power-on reset.



**EPFI**  
Bit 5

**Enable Power fail Interrupt.** This bit enables/disables the ability of the internal band-gap reference to generate a power-fail interrupt when  $V_{CC}$  falls below approximately 4.5 volts. While in Stop mode, both this bit and the Band-gap Select bit, BGS (EXIF.0), must be set to enable the power-fail interrupt.

0=Power-fail interrupt disabled.

1=Power-fail interrupt enabled during normal operation. Power-fail interrupt enabled in Stop mode if BGS is set.

**PFI**  
Bit 4

**Power fail Interrupt Flag.** When set, this bit indicates that a power-fail interrupt has occurred. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a power-fail interrupt, if enabled.

**WDIF**  
Bit 3

**Watchdog Interrupt Flag.** This bit, in conjunction with the Watchdog Timer Interrupt Enable bit, EWDI (EIE.4), and Enable Watchdog Timer Reset bit (WDCON.1), indicates if a watchdog timer event has occurred and what action will be taken. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a watchdog interrupt if enabled. This bit can only be modified using a Timed Access Procedure.

EWT	EWDI	WDIF	RESULT
x	x	0	No watchdog event has occurred.
0	0	1	Watchdog time-out has expired. No interrupt has been generated.
0	1	1	Watchdog interrupt has occurred.
1	0	1	Watchdog time-out has expired. No interrupt has been generated. Watchdog timer reset will occur in 512 cycles if RWT is not strobed.
1	1	1	Watchdog interrupt has occurred. Watchdog timer reset will occur in 512 cycles if RWT is not set using a Timed Access procedure.

**WTRF**  
Bit 2

**Watchdog Timer Reset Flag.** When set, this bit indicates that a watchdog timer reset has occurred. It is typically interrogated to determine if a reset was caused by watchdog timer reset. It is cleared by a power-on reset, but otherwise must be cleared by software before the next reset of any kind or software may erroneously determine that a watchdog timer reset has occurred. Setting this bit in software will not generate a watchdog timer reset. If the EWT bit is cleared, the watchdog timer will have no effect on this bit.

**EWT**  
Bit 1

**Enable Watchdog Timer Reset.** This bit enables/disables the ability of the watchdog timer to reset the device. This bit has no effect on the ability of the watchdog timer to generate a watchdog interrupt. The time-out period of the watchdog timer is controlled by the Watchdog Timer Mode Select bits (CKCON.7–6). Clearing this bit will disable the ability of the watchdog timer to generate a reset, but have no affect on the timer itself, or its ability to generate a watchdog timer interrupt. This bit can only be modified using a Timed Access Procedure. The default power-on reset state of this bit is 0 on the

ROMless devices. If the device contains EPROM program memory, the default power-on reset state of EWT is determined by the Watchdog Default POR State bit (WDPOR) located in the System Control Byte. If the device contains mask ROM program memory, the default power-on reset state is determined by a mask option.

0=A timeout of the watchdog timer will not cause the device to reset.

1=A timeout of the watchdog timer will cause the device to reset.

#### RWT

Bit 0

**Reset Watchdog Timer.** Setting this bit will reset the watchdog timer count.

This bit must be set using a Timed Access procedure before the watchdog timer expires, or a watchdog timer reset and/or interrupt will be generated if enabled. The time-out period is defined by the Watchdog Timer Mode Select bits (CKCON.7–6). This bit will always be 0 when read.

#### Accumulator (A or ACC)

	7	6	5	4	3	2	1	0
SFR E0h	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

#### ACC.7–0

Bits 7–0

**Accumulator.** This register serves as the accumulator for arithmetic operations. It is functionally identical to the accumulator found in the 80C32.

#### Extended Interrupt Enable (EIE)

	7	6	5	4	3	2	1	0
SFR E8h	–	–	ERTCI	EWDI	EX5	EX4	EX3	EX2
			RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset

Bit 7–6

Reserved. Read data will be indeterminate.

#### ERTCI

Bit 5

**Real Time Clock Interrupt Enable.** This bit enables/disables the real-time clock interrupt on the DS87C530. This bit will read 0 on all other devices.

0=Disable the real-time clock interrupt.

1=Enable interrupt requests generated by the real-time clock.

#### EWDI

Bit 4

**Watchdog Interrupt Enable.** This bit enables/disables the watchdog interrupt.

0=Disable the watchdog interrupt.

1=Enable interrupt requests generated by the watchdog timer.

#### EX5

Bit 3

**External Interrupt 5 Enable.** This bit enables/disables external interrupt 5.

0=Disable external interrupt 5.

1=Enable interrupt requests generated by the  $\overline{\text{INT5}}$  pin

#### EX4

Bit 2

**External Interrupt 4 Enable.** This bit enables/disables external interrupt 4.

0=Disable external interrupt 4.

1=Enable interrupt requests generated by the INT4 pin

**EX3**

Bit 1

**External Interrupt 3 Enable.** This bit enables/disables external interrupt 3.

0=Disable external interrupt 3.

1=Enable interrupt requests generated by the INT3 pin.

**EX2**

Bit 0

**External Interrupt 2 Enable.** This bit enables/disables external interrupt 2.

0=Disable external interrupt 2.

1=Enable interrupt requests generated by the INT2 pin.

**B Register (B)**

	7	6	5	4	3	2	1	0
SFR F0h	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

**B.7-0**

Bits 7-0

**B Register.** This register serves as a second accumulator for certain arithmetic operations. It is functionally identical to the B register found in the 80C32.**Real Time Alarm Subsecond Register (RTASS)**

	7	6	5	4	3	2	1	0
SFR F2h	RTASS.7	RTASS.6	RTASS.5	RTASS.4	RTASS.3	RTASS.2	RTASS.1	RTASS.0
	RW-*	RW-*	RW-*	RW-*	RW-*	RW-*	RW-*	RW-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

**RTASS.7-0**

Bits 7-0

**Real Time Alarm Subsecond.** These bits represent the subsecond alarm which will be compared against the RTC Subsecond register (RTCSS;FAh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Subsecond Register Compare Enable bit (RTCC.7). The contents of this register will be indeterminate following a no-battery reset, and unchanged by all other forms of reset.**Real Time Alarm Second Register (RTAS)**

	7	6	5	4	3	2	1	0
SFR F3h	0	0	RTAS.5	RTAS.4	RTAS.3	RTAS.2	RTAS.1	RTAS.0
	R-0	R-0	RW-*	RW-*	RW-*	RW-*	RW-*	RW-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

Bits 7-6

Reserved. These bits will be 0 when read.

**RTAS.5-0**

Bits 5-0

**Real Time Alarm Second.** These bits represent the second alarm which will be compared against the RTC Second register (RTCS;FBh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Second Register Compare Enable bit (RTCC.6). This register should only be loaded with values from 0 to 3Bh (0 to 59 seconds). The contents of this register will be indeterminate following a no-battery reset (except bits 7, 6), and unchanged by all other forms of reset.

**Real Time Alarm Minute Register (RTAM)**

	7	6	5	4	3	2	1	0
SFR F4h	0	0	RTAM.5	RTAM.4	RTAM.3	RTAM.2	RTAM.1	RTAM.0
	R-0	R-0	RW-*	RW-*	RW-*	RW-*	RW-*	RW-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

Bits 7-6

Reserved. These bits will be 0 when read.

**RTAM.5-0**

Bits 5-0

**Real Time Alarm Minute.** These bits represent the minute alarm which will be compared against the RTC Minute register (RTCM;FCh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Minute Register Compare Enable bit (RTCC.5). ). This register should only be loaded with values from 0 to 3Bh (0 to 59 minutes). The contents of this register will be indeterminate following a no-battery reset (except bits 7, 6), and unchanged by all other forms of reset.

**Real Time Alarm Hour Register (RTAH)**

	7	6	5	4	3	2	1	0
SFR F5h	0	0	0	RTAH.4	RTAH.3	RTAH.2	RTAH.1	RTAH.0
	R-0	R-0	R-0	RW-*	RW-*	RW-*	RW-*	RW-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

Bits 7-5

Reserved. These bits will be 0 when read.

**RTAH.4-0**

Bits 4-0

**Real Time Alarm Hour.** These bits represent the hour alarm which will be compared against the RTC Hour register (RTCH;FDh). The ability of a match between the two registers to cause an alarm is controlled by the RTC Hour Register Compare Enable bit (RTCC.4). This register should only be loaded with values from 0 to 17h (0 to 23 hours). The day of week bits DOW2-0, located in RTCH.7-5 do not have a corresponding alarm feature. The contents of this register will be indeterminate following a no-battery reset (except bits 7, 6, 5), and unchanged by all other forms of reset.

**Extended Interrupt Priority (EIP)**

	7	6	5	4	3	2	1	0
SFR F5h	-	-	PRTCI	PWDI	PX5	PX4	PX3	PX2
			RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset

Bit 7-6

Reserved. Read data will be indeterminate.

**PRTCI**

Bit 5

**Real Time Clock Interrupt Priority.** This bit controls the priority of the real-time clock interrupt on the DS87C530. This bit will read 0 on all other devices.

0=The real-time clock interrupt is a low priority interrupt.

1=The real-time clock interrupt is a high priority interrupt.

<b>PWDI</b> Bit 4	<b>Watchdog Interrupt Priority.</b> This bit controls the priority of the watchdog interrupt. 0=The watchdog interrupt is a low priority interrupt. 1=The watchdog interrupt is a high priority interrupt.
<b>PX5</b> Bit 3	<b>External Interrupt 5 Priority.</b> This bit controls the priority of external interrupt 5. 0=External interrupt 5 is a low priority interrupt. 1=External interrupt 5 is a high priority interrupt.
<b>PX4</b> Bit 2	<b>External Interrupt 4 Priority.</b> This bit controls the priority of external interrupt 4. 0=External interrupt 4 is a low priority interrupt. 1=External interrupt 4 is a high priority interrupt.
<b>PX3</b> Bit 1	<b>External Interrupt 3 Priority.</b> This bit controls the priority of external interrupt 3. 0=External interrupt 3 is a low priority interrupt. 1=External interrupt 3 is a high priority interrupt.
<b>PX2</b> Bit 0	<b>External Interrupt 2 Priority.</b> This bit controls the priority of external interrupt 2. 0=External interrupt 2 is a low priority interrupt. 1=External interrupt 2 is a high priority interrupt.

**Real Time Clock Control Register (RTCC)**

	7	6	5	4	3	2	1	0
SFR F9h	SSCE	SCE	MCE	HCE	RTCRE	RTCWE	RTCIF	RTCE
	RW-*	RW-*	RW-*	RW-*	RW*-0	RT*-0	R*-*	RT-*

R=Unrestricted Read, W=Unrestricted Write, T=Timed Access Write Only, -n=Value after Reset,  
\*=See description

<b>SSCE</b> Bit 7	<b>RTC Subsecond Register Compare Enable.</b> This bit enables a match between the Real Time Alarm Subsecond Register (RTASS;F2h) and the Real Time Clock Subsecond Register (RTCSS;FAh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no-battery reset, and is unaffected by all other resets. 0=The subsecond value is a Don't Care when evaluating the RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per subsecond tick (1/256 second) for as long as the other registers match. 1=Include the subseconds along with any other registers when evaluating alarm compare conditions.
<b>SCE</b> Bit 6	<b>RTC Second Register Compare Enable.</b> This bit enables a match between the Real Time Alarm Second Register (RTAS;F3h) and the Real Time Clock Second Register (RTCS;FBh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no-battery reset, and is unaffected by all other resets. 0=The second value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per second as long as the other registers match. 1=Include the second along with any other registers when evaluating alarm compare conditions.



**MCE**  
Bit 5

**RTC Minute Register Compare Enable.** This bit enables a match between the Real Time Alarm Minute Register (RTAM;F4h) and the Real Time Clock Minute Register (RTCM;FCh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no-battery reset, and is unaffected by all other resets.

0=The minute value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per minute as long as the other registers match.

1=Include the minute along with any other registers when evaluating alarm compare conditions.

**HCE**  
Bit 4

**RTC Hour Register Compare Enable.** This bit enables a match between the Real Time Alarm Hour Register (RTAH;F5h) and the Real Time Clock Hour Register (RTCH;FDh) to contribute to the RTC interrupt request. This bit will be indeterminate following a no-battery reset, and is unaffected by all other resets.

0=The hour value is a Don't Care when evaluating an RTC alarm. If any other alarm register compare bits are enabled, this will cause one interrupt per hour for as long as the other registers match.

1=Include the hour along with any other registers when evaluating alarm compare conditions.

**RTCRE**  
Bit 3

**RTC Read Enable.** This bit temporarily halts internal updating of the RTC to allow software to read the current time. No loss of time will occur. This bit will be cleared to 0 following any reset. Attempts to set the RTCRE and RTCWE bits simultaneously will be ignored. When this bit is cleared, software must wait 4 machine cycles before setting either the RTCRE or RTCWE bit again. 0=Reads of the RTC clock registers (RTCSS;FAh, RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh) are prohibited and will return erroneous values.

1=Reads of the RTC clock registers are permitted during a 1 ms window starting from the time the bit is set. Immediately after setting this bit, software must wait 4 machine cycles to allow all time registers to synchronize. This bit should be cleared by the user when the desired reads are complete, although it will clear automatically within 1.95 ms if not cleared in software.

**RTCWE**  
Bit 2

**RTC Write Enable.** This bit temporarily halts the RTC to allow software to update the current time. No loss of time will occur. This bit can only be modified using a Timed Access procedure. Changing this bit from 1 to 0 will reset the RTCSS register to 00h. This bit will be cleared to 0 following any reset. 0=Writes to the RTC clock registers (RTCSS;FAh, RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh) are ignored. Attempts to set the RTCRE and RTCWE bits simultaneously will be ignored. When this bit is cleared, software must wait 4 machine cycles before setting either the RTCRE or RTCWE bit again.

1=Writes to the RTC clock registers are permitted during a 1 ms window starting from the time this bit is set. Immediately after setting this bit, software must wait 4 machine cycles to allow all time registers to synchronize. This bit should be cleared by the user when the desired updates are complete, although it will clear automatically after 1.95 ms if not cleared in software.

**RTCIF**

Bit 1

**RTC Interrupt Flag.** This bit indicates that a RTC alarm match has been made between all the enabled alarm registers and their corresponding clock registers. This bit will generate an RTC Interrupt if the ERTCI bit (EIE.5) is set, and must be cleared by software following an interrupt. RTC interrupts cannot be generated by setting this bit. Clearing all alarm compare enable bits (RTCC.7–4) will also clear this bit. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets. This bit cannot be set in software.

0=No RTC interrupts are pending.

1=RTC Interrupt is pending/active.

**RTCE**

Bit 0

**RTC Enable.** This bit enables/disables the RTC oscillator, halting the RTC. This bit must be accessed using a Timed Access procedure. This bit will be indeterminate following a no–battery reset, and is unaffected by all other resets. If RTC operation is desired, it must be enabled following battery application.

0=RTC oscillator is disabled.

1=RTC oscillator is enabled.

**Real Time Clock Subsecond Register (RTCSS)**

	7	6	5	4	3	2	1	0
SFR F4h	RTCSS.7	RTCSS.6	RTCSS.5	RTCSS.4	RTCSS.3	RTCSS.2	RTCSS.1	RTCSS.0
	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*	R*–*

R=Unrestricted Read, –n=Value after Reset, \*=See description

**RTCSS.7–0**

Bits 7–0

**Real Time Clock Subseconds.** This register represents the subsecond value of the RTC. It can be read only when the RTCRE bit is set, and writes are not permitted. It is reset to 00h when the RTCWE bit is cleared. The register counts from 0h to FFh.

**Real Time Clock Second Register (RTCS)**

	7	6	5	4	3	2	1	0
SFR FBh	0	0	RTCS.5	RTCS.4	RTCS.3	RTCS.2	RTCS.1	RTCS.0
	R*–0	R*–0	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*	R*W*–*

R=Unrestricted Read, W=Unrestricted Write, –n=Value after Reset, \*=See description

Bits 7–6

Reserved. These bits will be 0 when read.

**RTCS.5–0**

Bits 5–0

**Real Time Clock Seconds.** This register represents the second value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 3Bh (0 to 59 seconds), and any writes to this register outside of that range will generate an inaccurate count.

**Real Time Clock Minute Register (RTCM)**

	7	6	5	4	3	2	1	0
SFR FCh	0	0	RTCM.5	RTCM.4	RTCM.3	RTCM.2	RTCM.1	RTCM.0
	R*-0	R*-0	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

Bits 7-6

Reserved. These bits will be 0 when read.

**RTCM.5-0**

Bits 5-0

**Real Time Clock Minutes.** This register represents the minute value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 3Bh (0 to 59 minutes), and any writes to this register outside of that range will generate an inaccurate count.

**Real Time Clock Hour Register (RTCH)**

	7	6	5	4	3	2	1	0
SFR FDh	DOW2	DOW1	DOW0	RTCH.4	RTCH.3	RTCH.2	RTCH.1	RTCH.0
	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*	R*W*-*

R=Unrestricted Read, W=Unrestricted Write, -n=Value after Reset, \*=See description

**DOW2-0**

Bits 7-5

**Real Time Clock Day of the Week.** These bits represent the current day of the week. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 1h to 7h, and increments when the hour value of the RTC (RTCH.4-0) rolls over from 17h to 0h. Writing a 0h to these bits will disable the day of week function and the count will remain 0. No alarm corresponds to these bits.

**RTCH.4-0**

Bits 4-0

**Real Time Clock Hours.** These bits represent the hour value of the RTC. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. This register counts from 0h to 17h (0 to 23 hours), and any writes outside of that range will generate an inaccurate count.

**Real Time Clock Day Register 0 (RTCD0)**

	7	6	5	4	3	2	1	0
SFR FEh	RTCD0.7	RTCD0.6	RTCD0.5	RTCD0.4	RTCD0.3	RTCD0.2	RTCD0.1	RTCD0.0
	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*

R=Unrestricted Read, W=Unrestricted Write, ~n=Value after Reset, \*=See description

**RTCD0.7–0**  
Bits 7–0

**Real Time Clock Day Register 0.** This register contains the least significant byte of the 16-bit current day count. This is not an absolute value tied to a specific calendar date, but rather a relative day count defined by the user. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. The register counts from 0h to FFh. No alarm corresponds to these bits.

**Real Time Clock Day Register 1 (RTCD1)**

	7	6	5	4	3	2	1	0
SFR FFh	RTCD1.7	RTCD1.6	RTCD1.5	RTCD1.4	RTCD1.3	RTCD1.2	RTCD1.1	RTCD1.0
	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*	R*W*~*

R=Unrestricted Read, W=Unrestricted Write, ~n=Value after Reset, \*=See description

**RTCD1.7–0**  
Bits 7–0

**Real Time Clock Day Register 1.** This register contains the most significant byte of the 16-bit current day count. This is not an absolute value tied to a specific calendar date, but rather a relative day count defined by the user. This register can be read only when the RTCRE bit is set, and can only be modified when the RTCWE bit is set. Consult the description of the RTCWE bit for the programming protocol for this register. The register counts from 0h to FFh. A rollover of this register will clear RTCD1 and RTCD0. No alarm corresponds to these bits.

## INSTRUCTION TIMING

All instructions in the High-Speed Microcontroller perform the same functions as their 80C32 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute terms of nanoseconds for a given crystal, and in relative terms of clocks.

For absolute timing of real-time events, the timing of software loops will need to be calculated using the data provided in Section 16, Instruction Set Details. However, timers default to run at the older 12 clocks per timer increment. Therefore, while software runs at higher speed, timer-based events need no modification.

The relative time of two instructions might be different in the new architecture than it was previously. For example, both the one-byte, two-cycle "MOVX A, @DPTR" instruction and the three-byte, two-cycle "MOV direct, direct" instruction used two cycles. Therefore, they required the same amount of time. In the High-Speed Microcontroller, the MOVX instruction uses two cycles but the "MOV direct, direct" uses three cycles. While both are faster than their original counterparts, they now have different execution times from each other. This is because the High-Speed Microcontroller typically uses one cycle for each byte. This is generally true for all instructions except for MUL, DIV, MOVC, MOVX, and branch type instructions. These require more cycles than the number of bytes involved. The timing of each instruction should be examined for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV.

## ADDRESSING MODES

The High-Speed Microcontroller uses the standard 8051 instruction set which is supported by a wide range of third party assemblers and compilers. Like the 8051, the High-Speed Microcontroller uses three memory areas. These are program memory, data memory, and

Registers. Both the program and data areas are 64KB each. They extend from 0000h to FFFFh. The register areas are located between 00h and FFh, but do not overlap with the program and data segments. This is because the High-Speed Microcontroller uses different modes of addressing to reach each memory segment. These modes are described below.

Program memory is the area from which all instructions are fetched. It is inherently read only. This is because the 8051 instruction set provides no instructions that write to this area. Read/write access is for data memory and Registers only. No special action is required to fetch from program memory. Each instruction fetch will be performed automatically by the on-chip hardware. In versions that contain on-chip memory, the hardware will decide whether the fetch is on-chip or off-chip based on the address.

Explicit addressing modes are needed for the data memory and register areas. These modes determine which register area is accessed or if off-chip data memory is used.

The High-Speed Microcontroller supports eight addressing modes. They are:

- Register Addressing
- Direct Addressing
- Register Indirect Addressing
- Immediate Addressing
- Register Indirect Addressing with Displacement
- Relative Addressing
- Page Addressing
- Extended Addressing

Five of the eight are used to address operands. The remainder are used for program control and branching. When writing assembly language instructions that use arguments, the convention is destination, source. Each mode of addressing is summarized below. Note that many instructions (such as ADD) have multiple addressing modes available..



## Register Addressing

Register Addressing is used for operands that are located in one of the eight Working Registers (R7–R0). These are the currently selected Working Register bank, which reside in the lower 32 bytes of Scratchpad RAM. A register bank is selected using two bits in the Program Status Word (PSW;D0h). This addressing mode is powerful, since it uses the active bank without knowing which bank is selected. Thus one instruction can have multiple uses by simply switching banks. Register Addressing is also a high-speed instruction, requiring only one machine cycle. Two examples of Register Addressing are provided below.

```
ADD    A, R4    ;Add Accumulator to register R4
INC     R2      ;Increment the value in register R2
```

In the first case, the value in R4 is the source of the operation. In the later, R2 is the destination. These instructions do not consider the absolute address of the register. They will act on whichever bank has been selected.

Any Working Register may also be accessed by Direct Addressing, described below. To do this, the absolute address must be specified.

## Direct Addressing

Direct Addressing is the mode used to access the entire lower 128 bytes of Scratchpad RAM and the SFR area. It is commonly used to move the value in one register to another. Two examples are shown below.

```
MOV     72h, 74h ;Move the value in register 74 to
                ; register 72.
MOV     90h, 20h ;Move the value in register 20 to
                ; the SFR at 90h (Port 1)
```

Note that there is no instruction difference between a RAM access and an SFR access. The SFRs are simply register locations above 7Fh.

Direct Addressing also extends to bit addressing. There is a group of instructions that explicitly use bits. The address information provided to such an instruction is the bit location, rather than the register address. Registers between 20h and 2Fh contain bits that are individually addressable. SFRs that end in 0 or 8 are bit

addressable. An example of Direct Bit Addressing is as follows.

```
SETB    00h      ;Set bit 00 in the RAM. This is the
                ; LSB of the register at address 20h
                ; as shown in Section 4.
MOV     C, 0B7h  ;Move the contents of bit B7 to the
                ; Carry flag. Bit B7 is the MSB of
                ; register B0 (Port 3).
```

## Register Indirect Addressing

This mode is used to access the Scratchpad RAM locations above 7Fh. It can also be used to reach the lower RAM (0h – 7Fh) if needed. The address is supplied by the contents of the Working Register specified in the instruction. Thus one instruction can be used to reach many values by altering the contents of the designated Working Register. Note that in general, only R0 and R1 can be used as pointers. An example of Register Indirect Addressing is as follows.

```
ANL     A, @R0   ;Logical AND the Accumulator
                ; with the contents of the register
                ; pointed to by the value stored in
                ; R0.
```

This mode is also used for Stack manipulation. This is because all Stack references are directed by the value in the Stack Pointer register. The Push and Pop instructions use this method of addressing. An example is as follows.

```
PUSH    A        ;Saves the contents of the
                ; accumulator on the stack.
```

Register Indirect Addressing is used for all off-chip data memory accesses. These involve the MOVX instruction. The pointer registers can be R0, R1, DPTR0 and DPTR1. Both R0 and R1 reside in the Working Register area of the Scratchpad RAM. They can be used to reference a 256 byte area of off-chip data memory. When using this type of addressing, the upper address byte is supplied by the value in the Port 2 latch. This value must be selected by software prior to the MOVX instruction. An example is as follows.

```
MOVBX   @R0, A   ;Write the value in the accumulator
                ; to the address pointed to by R0 in
                ; the page pointed to by P2.
```

The 16-bit Data pointers (DPTRs) can be used as an absolute off-chip reference. This gives access to the entire 64KB data memory map. An example is as follows.

```
MOVX @DPTR, A ;Write the value in the accumulator
                ; to the address referenced by the
                ; selected data pointer.
```

### Immediate Addressing

Immediate Addressing is used when one of the operands is predetermined and coded into the software. This mode is commonly used to initialize SFRs and to mask particular bits without affecting others. An example is as follows.

```
ORL A, #40h ;Logical OR the Accumulator
            ; with 40h.
```

### Register Indirect with Displacement

Register Indirect Addressing with Displacement is used to access data in lookup tables in program memory space. The location is created using a base address with an index. The base address can be either the PC or the DPTR. The index is the accumulator. The result is stored in the accumulator. An example is as follows.

```
MOVC A, @A+DPTR ;Load the accumulator with the
                ; contents of program memory
                ; pointed to by the contents of
                ; the DPTR plus the value in
                ; the accumulator.
```

### Relative Addressing

Relative Addressing is used to determine a destination address for Conditional branch. Each of these instructions includes an 8-bit value that contains a two's com-

plement address offset (–127 to +128) which is added to the PC to determine the destination address. This destination is branched to when the tested condition is true. The PC points to the program memory location immediately following the branch instruction when the offset is added. If the tested condition is not true, the next instruction is performed. An example is as follows.

```
JZ $-20 ;Branch to the location (PC+2)–20
        ; if the contents of the accumulator
        ;=0.
```

### Page Addressing

Page Addressing is used by the Branching instructions to specify a destination address within the same 2KB block as the next contiguous instruction. The full 16-bit address is calculated by taking the five highest order bits for the next instruction (PC+2) and concatenating them with the lowest order 11 bit field contained in the current instruction. An example is as follows.

```
0870h ACALL100h ;Call to the subroutine at
                ; address 100h plus the
                ; current page address.
```

In this example, the current page address is 800h, so the destination address is 900h.

### Extended Addressing

Extended Addressing is used by the Branching instructions to specify a 16-bit destination address within the 64KB address space. The destination address is fixed in software as an absolute value. An example is as follows.

```
LJMP 0F732h ; Jump to address 0F732h.
```

**PROGRAM STATUS FLAGS**

All Program Status Flags are contained in the Program Status Word at SFR location D0h. It contains flags that

reflect the status of the CPU and the result of selected operations. The flags are summarized below. The following table shows the instructions that affect each flag.

**Bit Description :****PSW.7**

Carry

**C**

Set when the previous operation resulted in a carry (during addition) or a borrow (during subtraction), otherwise cleared.

**PSW.6**

Auxiliary Carry

**AC**

Set when the previous operation resulted in a carry (during addition) or a borrow (during subtraction) from the high order nibble. Otherwise cleared.

**PSW.2**

Overflow

**OV**

Set when a carry was generated into the high order bit but not a carry out of the high order bit. OV is normally used with 2's complement arithmetic.

**PSW.0**

Parity

**P**

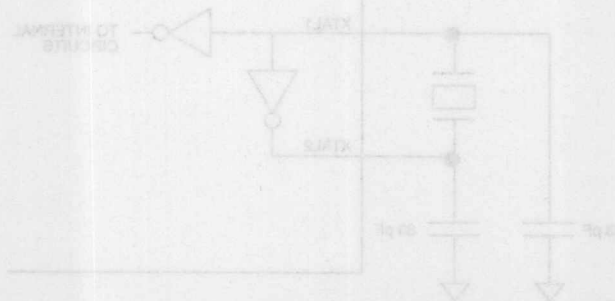
Set to logic 1 to indicate an odd number of ones in the accumulator (odd parity). Cleared for an even number of ones. This produces even parity.

All of these bits are cleared to a logic 0 for all resets.

**INSTRUCTIONS THAT AFFECT FLAG SETTINGS** Table 4-4

INSTRUCTION	FLAGS			INSTRUCTION	FLAGS		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C, $\overline{\text{bit}}$	X		
DIV	0	X		ORL C, bit	X		
DA	X			ORL C, $\overline{\text{bit}}$	X		
RRC	X			MOV C, bit	X		
RLC	X			CJNE	X		
SETB C	1						

X indicates the modification is according to the result of the instruction.



**SECTION 5: CPU TIMING**

The timing of the High-Speed Microcontroller is the area with the greatest departure from the original 8051 series. This section will briefly explain the timing and also compare it to the original.

**OSCILLATOR**

The High-Speed Microcontroller provides an on-chip oscillator circuit that can be driven by an external crystal or by an off-chip TTL clock source. The oscillator circuit provides the internal clocking signals to the on-chip CPU and I/O circuits.

Figure 5-1 shows the required connections for a crystal. In most cases, a crystal will be the preferred clock source. For very low power applications, a low frequency ceramic resonator may also be used. The capacitors shown in Figure 5-1 are typical values. If a resonator is used, higher capacitance, such as 47 pF may be needed.

For higher frequency designs, an off-chip clock oscillator may be preferred. This is illustrated in Figure 5-2. When using an off-chip oscillator, the duty cycle becomes important. As nearly as possible, a 50% duty cycle should be supplied.

**XTAL1**

This pin is the input to an inverting high gain amplifier. It also serves as the input for an off-chip oscillator. Note that when using an off-chip oscillator, XTAL2 is disconnected.

**XTAL2**

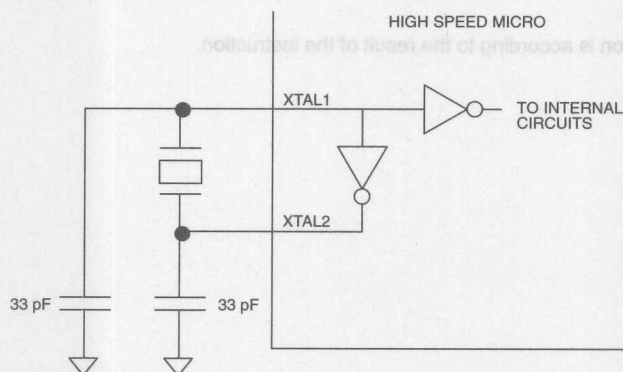
This pin is the output of the crystal amplifier. It can be used to distribute the clock to other devices on the same board. If using a crystal, the loading on this pin should be kept to a minimum, especially capacitive loading.

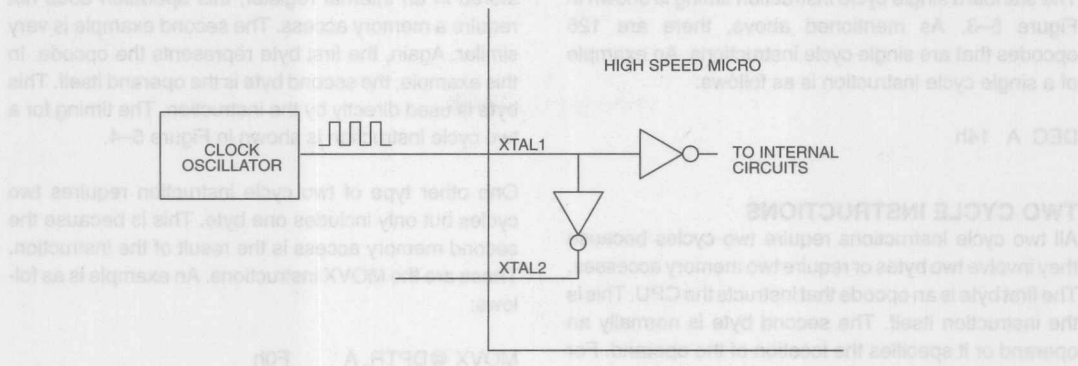
**OSCILLATOR CHARACTERISTICS**

The High-Speed Microcontroller was designed to operate with a parallel resonant AT cut crystal. The crystal should resonate at the desired frequency in its primary or fundamental mode. The oscillator employs a high gain amplifier to assure a clean waveform at high frequency. Due to the high performance nature of the product, both clock edges are used for internal timing. Therefore, the duty cycle of the clock source is of importance. A crystal circuit will balance itself automatically. Thus crystal users will not need to take extra precautions concerning duty cycle.

**CRYSTAL SELECTION**

The High-Speed Microcontroller family was designed to operate with fundamental mode crystals for improved stability. Although most high speed (i.e. greater than 25 MHz) crystals operate from their third overtone, fundamental mode crystals are available from most major crystal suppliers. Designers are cautioned to ensure that high speed crystals being specified for use in their application do operate at the rated frequency in their fundamental mode. The use of a third overtone crystal will typically result in oscillation rates at one-third the desired speed.

**CRYSTAL CONNECTION** Figure 5-1

**CLOCK SOURCE INPUT** Figure 5-2**INSTRUCTION TIMING**

The clock source, whether crystal or oscillator, supplies the internal functions with a precise time base. The clock is used to create the basic unit of timing called a machine cycle. One machine cycle consists of four clocks when operating in divide by 4 mode. The use of Power Management modes will cause the device to utilize 64 or 1024 external clock cycles per machine cycle. Each clock within a machine cycle produces a state designated with a C. Thus, within a machine cycle there are four states called C1, C2, C3, and C4. Various operations take place during each C state. Within this section and throughout others, an event timing will be identified by its C state. For example, ALE rises at the beginning of the C1 time. Since the clock source is the source of nearly all timing, the electrical specifications are given in terms of clocks. The time of a clock period is referred to as  $t_{CLCL}$ .

Most times in the electrical specifications are specified as some number of clocks from the edge of a signal. The signal edges were also derived from the clock source and the C states.

Due to the limited number of edges within a machine cycle, selected events must occur between edges. The High-Speed Microcontroller employs sophisticated circuits to create half and quarter clock events. That is, some events occur between clock edges. Such circuits assure that events occur as precisely as if a clock edge were available. While being generally transparent to the user, these circuits result in the use of fractional clocks in the electrical specifications. For example, a time may be specified as  $2.5 t_{CLCL}$ .

As mentioned above, a machine cycle is the basic timing unit of most functions in the High-Speed Microcontroller. A machine cycle of the High-Speed Microcontroller is the time required to execute a single cycle instruction. Almost half the opcodes (126 of 255) of the 8051 instruction set are implemented in a single machine cycle in the High-Speed Microcontroller. The remaining instructions require multiple machine cycles.

The Power Management Modes implemented on some devices modify the number of clock cycles needed to execute an instruction. Instead of 4 clocks per machine cycle, power management mode 1 (PMM1) and power management mode 2 (PMM2) utilize 64 and 1024 clocks per cycle respectively to conserve power. A full description of the power management modes and their effect on CPU operation is provided in Section 7.

All instructions are coded within an 8-bit field called an opcode. This single byte must be fetched from program memory. The opcode is decoded by the CPU. It determines what action the microcontroller will take and whether more information is needed from memory. If no other memory is needed, then only one byte was required. Thus the instruction is called a one byte instruction. In some cases, more data is needed. These will be two or three byte instructions.

In most cases, the number of memory accesses (bytes) needed by an instruction is equal to the number of machine cycles. Thus single cycle instructions contain one byte, and two cycle instructions have two bytes. This is true except for the special cases mentioned below.



## SINGLE CYCLE INSTRUCTIONS

The standard single cycle instruction timing is shown in Figure 5-3. As mentioned above, there are 126 opcodes that are single cycle instructions. An example of a single cycle instruction is as follows:

DEC A 14h

## TWO CYCLE INSTRUCTIONS

All two cycle instructions require two cycles because they involve two bytes or require two memory accesses. The first byte is an opcode that instructs the CPU. This is the instruction itself. The second byte is normally an operand or it specifies the location of the operand. For example, the instruction "ANL A, direct" uses two cycles and requires two bytes. Two examples are as follows:

ANL A, direct      55h  
                         a7-0

ANL A, #data      54h  
                         d7-d0

Note that in the first example, the first memory access is the location of the opcode. The second memory access is the location

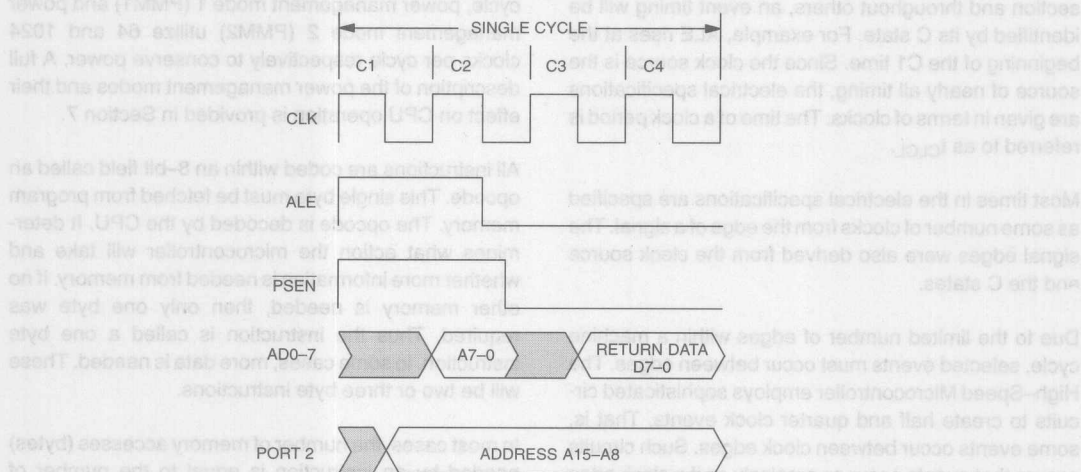
of the operand in the register map. Since the result is stored in an internal register, this operation does not require a memory access. The second example is very similar. Again, the first byte represents the opcode. In this example, the second byte is the operand itself. This byte is used directly by the instruction. The timing for a two cycle instruction is shown in Figure 5-4.

One other type of two cycle instruction requires two cycles but only includes one byte. This is because the second memory access is the result of the instruction. These are the MOVX instructions. An example is as follows:

MOVX @DPTR, A      F0h

The second cycle in this instruction is the write to data memory at the address pointed to by the data pointer. Thus this instruction is a two cycle one byte instruction, but requires two memory accesses. The MOVX timing is a special case, since the user can control it with the Stretch MOVX feature. The timing for the Stretch MOVX is discussed in the section on Memory Access.

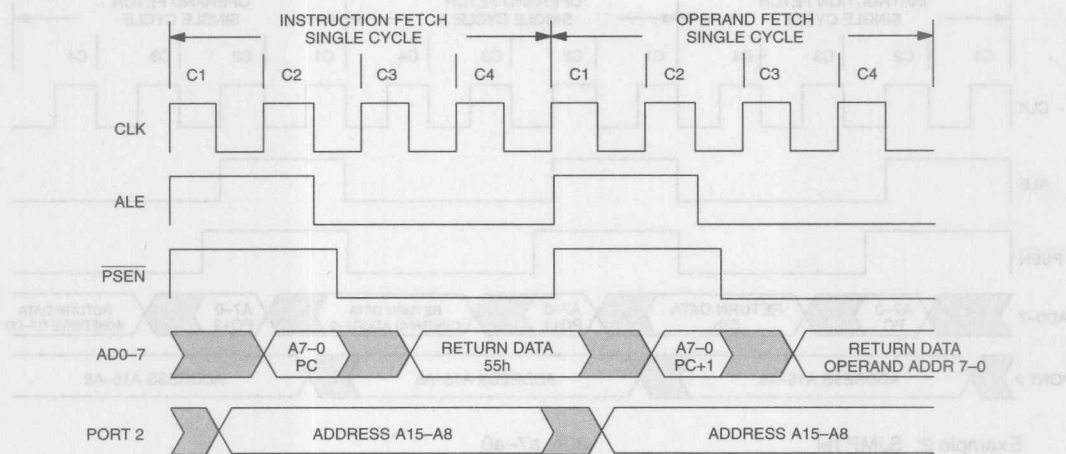
**SINGLE CYCLE INSTRUCTION TIMING** Figure 5-3



\*Shaded areas are held in a weak latch on the port until overdriven.

**TWO CYCLE INSTRUCTION TIMING** Figure 5-4

Example: ANL A, direct : 55h addr7-0



\*Shaded areas are held in a weak latch on the port until overdriven.

**THREE CYCLE INSTRUCTIONS**

Three cycle instructions come in two varieties. The first requires three memory accesses. These are similar to one and two cycle instructions in that the number of bytes equals the number of cycles.

The second variety is a three cycle instruction that simply requires 12 clocks to perform the function. This may have one or two bytes. Examples of both types are shown below.

ANL direct, #data	53h	(3 bytes)
	a7-a0	
	d7-d0	
SJMP rel	80h	(2 bytes)
	a7-a0	
INC DPTR	A3h	(1 byte)

In the first example, the first memory fetch is the opcode. The second is the location of the destination register. The third memory fetch is the operand that is used by the instruction. This instruction has three memory accesses, so it requires three machine cycles. The second example has the operand in the first byte and the jump location in the second. It requires three cycles to actually perform the jump. The third example contains simply the opcode, which is one byte. This instruction involves the manipulation of a 16-bit register so it takes

longer than 8-bit operations. Figure 5-5 shows the timing of all three types of three cycle instructions.

**FOUR CYCLE INSTRUCTIONS**

All four cycle instructions require more time than the associated number of bytes. These are all program branching instructions that can move program control to a new location. The four cycle instructions use either 1 or 3 bytes as shown in the following examples. Figure 5-6 shows the timing of both four cycle instructions.

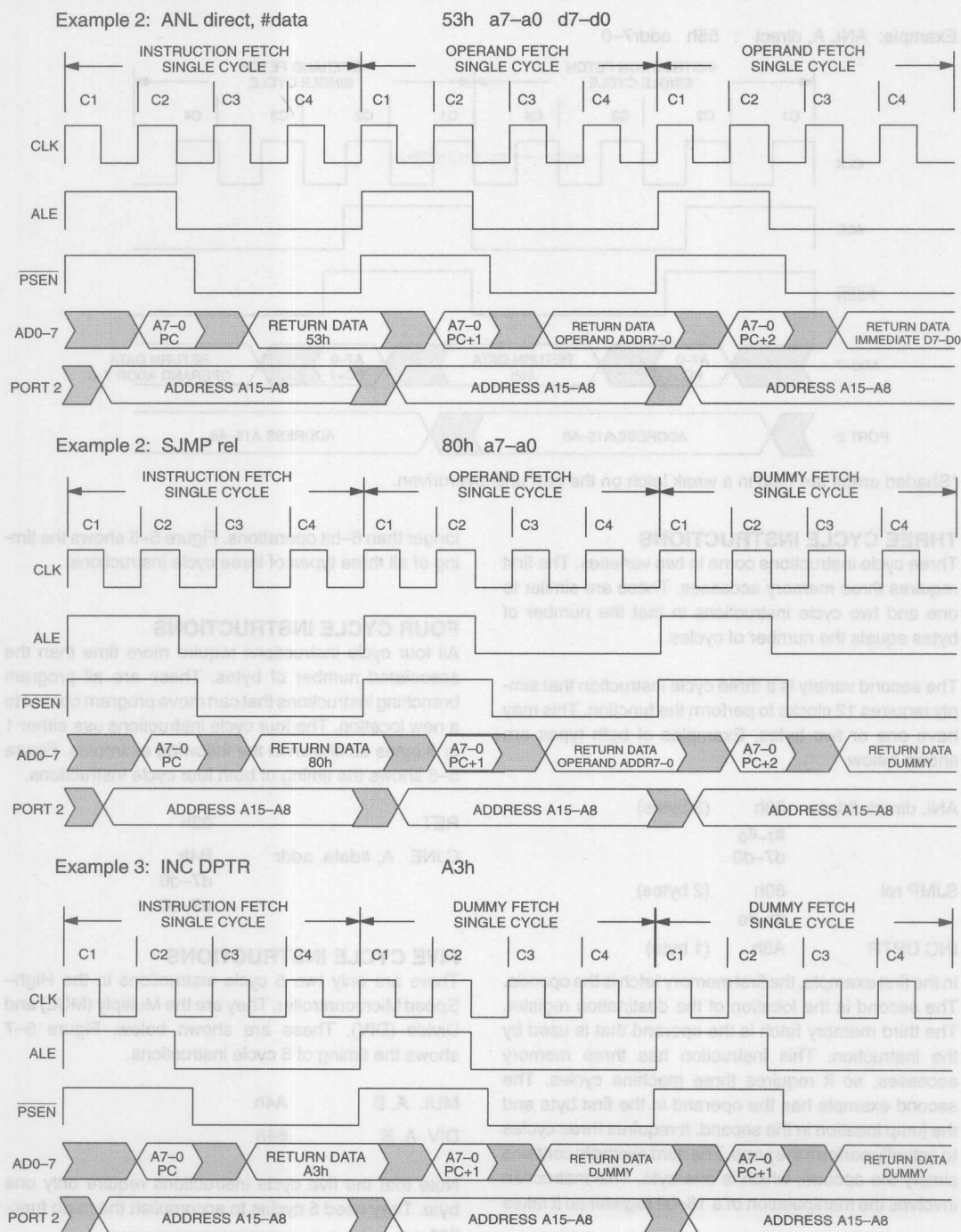
RET	22h
CJNE A, #data, addr	B4h
	d7-d0
	a7-a0

**FIVE CYCLE INSTRUCTIONS**

There are only two 5 cycle instructions in the High-Speed Microcontroller. They are the Multiply (MUL) and Divide (DIV). These are shown below. Figure 5-7 shows the timing of 5 cycle instructions.

MUL A, B	A4h
DIV A, B	84h

Note that the five cycle instructions require only one byte. They need 5 cycles to accomplish the math function.

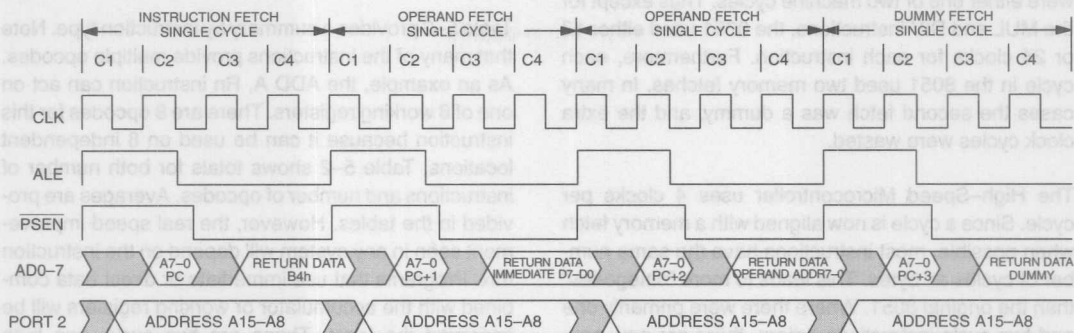
**THREE CYCLE INSTRUCTION TIMING** Figure 5-5

\*Shaded areas are held in a weak latch on the port until overdriven.

**FOUR CYCLE INSTRUCTION TIMING** Figure 5-6

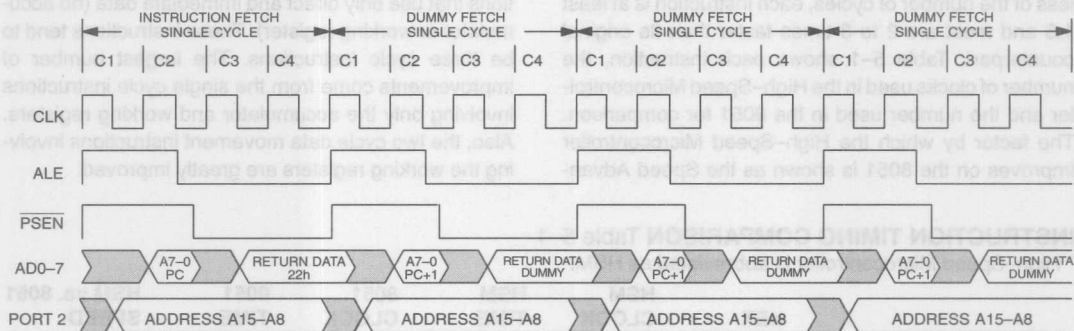
Example 1: CJNE A, #data, addr

B4h d7-d0 a7-a0



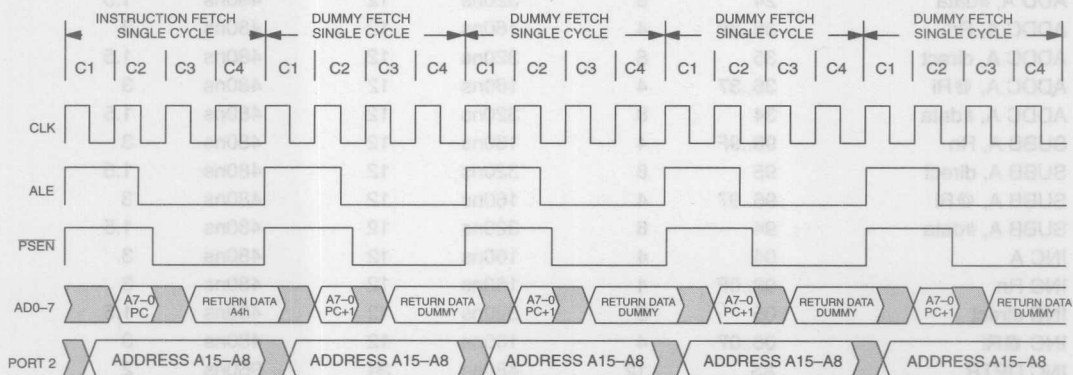
Example 2: RET

22h

**FIVE CYCLE INSTRUCTION TIMING** Figure 5-7

Example: MUL A,B

A4h



\*Shaded areas are held in a weak latch on the port until overdriven.

## COMPARISON TO THE 8051

The original 8051 had a 12 clock architecture. A machine cycle needed 12 clocks and most instructions were either one or two machine cycles. Thus except for the MUL and DIV instructions, the 8051 used either 12 or 24 clocks for each instruction. Furthermore, each cycle in the 8051 used two memory fetches. In many cases the second fetch was a dummy, and the extra clock cycles were wasted.

The High-Speed Microcontroller uses 4 clocks per cycle. Since a cycle is now aligned with a memory fetch when possible, most instructions have the same number of cycles as bytes. This leads to more "categories" than the original 8051. Where there were primarily one and two cycle instructions before, there are now one, two, three, and four cycle instructions. Multiply and Divide require five cycles. Note however, that regardless of the number of cycles, each instruction is at least 1.5 and most are 2 to 3 times faster than its original counterpart. Table 5-1 shows each instruction, the number of clocks used in the High-Speed Microcontroller and the number used in the 8051 for comparison. The factor by which the High-Speed Microcontroller improves on the 8051 is shown as the Speed Advan-

tage. A Speed Advantage of 3.0 means that the High-Speed Microcontroller performs the same instruction three times faster than the 8051.

Table 5-2 provides a summary by instruction type. Note that many of the instructions provide multiple opcodes. As an example, the ADD A, Rn instruction can act on one of 8 working registers. There are 8 opcodes for this instruction because it can be used on 8 independent locations. Table 5-2 shows totals for both number of instructions and number of opcodes. Averages are provided in the tables. However, the real speed improvement seen in any system will depend on the instruction mix. Programs that use immediate or direct data combined with the accumulator or working registers will be improved the least. These are two cycle, two byte instructions. Moderate performance improvement will be gained by emphasizing short branches and instructions that use only direct and immediate data (no accumulator or working register). These instructions tend to be three cycle instructions. The largest number of improvements come from the single cycle instructions involving only the accumulator and working registers. Also, the two cycle data movement instructions involving the working registers are greatly improved.

## INSTRUCTION TIMING COMPARISON Table 5-1

\* High-Speed Microcontroller is abbreviated as HSM.

INSTRUCTION	HEX CODE	HSM CLOCK CYCLES	HSM TIME @ 25 MHz	8051 CLOCK CYCLES	8051 TIME @25 MHz	HSM vs. 8051 SPEED ADVANTAGE
ADD A, Rn	28..2F	4	160ns	12	480ns	3
ADD A, direct	25	8	320ns	12	480ns	1.5
ADD A, @Ri	26..27	4	160ns	12	480ns	3
ADD A, #data	24	8	320ns	12	480ns	1.5
ADDC A, Rn	38..3F	4	160ns	12	480ns	3
ADDC A, direct	35	8	320ns	12	480ns	1.5
ADDC A, @Ri	36..37	4	160ns	12	480ns	3
ADDC A, #data	34	8	320ns	12	480ns	1.5
SUBB A, Rn	98..9F	4	160ns	12	480ns	3
SUBB A, direct	95	8	320ns	12	480ns	1.5
SUBB A, @Ri	96..97	4	160ns	12	480ns	3
SUBB A, #data	94	8	320ns	12	480ns	1.5
INC A	04	4	160ns	12	480ns	3
INC Rn	08..0F	4	160ns	12	480ns	3
INC direct	05	8	320ns	12	480ns	1.5
INC @Ri	06..07	4	160ns	12	480ns	3
INC DPTR	A3	12	480ns	24	960ns	2
DEC A	14	4	160ns	12	480ns	3
DEC Rn	18..1F	4	160ns	12	480ns	3
DEC direct	15	8	320ns	12	480ns	1.5



DEC @Ri	16..17	4	160ns	12	480ns	3
MUL AB	A4	20	800ns	48	1.92us	2.4
DIV AB	84	20	800ns	48	1.92us	2.4
DA A	D4	4	160ns	12	480ns	3
ANL A, Rn	58..5F	4	160ns	12	480ns	3
ANL A, direct	55	8	320ns	12	480ns	1.5
ANL A, @Ri	56..57	4	160ns	12	480ns	3
ANL A, #data	54	8	320ns	12	480ns	1.5
ANL direct, A	52	8	320ns	12	480ns	1.5
ANL direct, #data	53	12	480ns	24	960ns	2
ORL A, Rn	48..4F	4	160ns	12	480ns	3
ORL A, direct	45	8	320ns	12	480ns	1.5
ORL A, @Ri	46..47	4	160ns	12	480ns	3
ORL A, #data	44	8	320ns	12	480ns	1.5
ORL direct, A	42	8	320ns	12	480ns	1.5
ORL direct, #data	43	12	480ns	24	960ns	2
XRL A, Rn	68..6F	4	160ns	12	480ns	3
XRL A, direct	65	8	320ns	12	480ns	1.5
XRL A, @Ri	66..67	4	160ns	12	480ns	3
XRL A, #data	64	8	320ns	12	480ns	1.5
XRL direct, A	62	8	320ns	12	480ns	1.5
XRL direct, #data	63	12	480ns	24	960ns	2
CLR A	E4	4	160ns	12	480ns	3
CPL A	F4	4	160ns	12	480ns	3
RL A	23	4	160ns	12	480ns	3
RLC A	33	4	160ns	12	480ns	3
RR A	03	4	160ns	12	480ns	3
RRC A	13	4	160ns	12	480ns	3
SWAP A	C4	4	160ns	12	480ns	3
MOV A, Rn	E8..EF	4	160ns	12	480ns	3
MOV A, direct	E5	8	320ns	12	480ns	1.5
MOV A, @Ri	E6..E7	4	160ns	12	480ns	3
MOV A, #data	74	8	320ns	12	480ns	1.5
MOV Rn, A	F8..FF	4	160ns	12	480ns	3
MOV Rn, direct	A8..AF	8	320ns	24	960ns	3
MOV Rn, #data	78..7F	8	320ns	12	480ns	1.5
MOV direct, A	F5	8	320ns	12	480ns	1.5
MOV direct, Rn	88..8F	8	320ns	24	960ns	3
MOV direct, direct	85	12	480ns	24	960ns	2
MOV direct, @Ri	86..87	8	320ns	24	960ns	3
MOV direct, #data	75	12	480ns	24	960ns	2
MOV @Ri, A	F6..F7	4	160ns	12	480ns	3
MOV @Ri, direct	A6..A7	8	320ns	24	960ns	3
MOV @Ri, #data	76..77	8	320ns	12	480ns	1.5
MOV DPTR, #data16	90	12	480ns	24	960ns	2
MOVC A, @A+DPTR	93	12	480ns	24	960ns	2
MOVC A, @A+PC	83	12	480ns	24	960ns	2
MOVX A, @Ri	E2..E3	8	320ns	24	960ns	3
MOVX A, @DPTR	E0	8	320ns	24	960ns	3
MOVX @Ri, A	F2..F3	8	320ns	24	960ns	3

MOVX @DPTR, A	F0	8	320ns	24	960ns	3
PUSH direct	C0	8	320ns	24	960ns	3
POP direct	D0	8	320ns	24	960ns	3
XCH A, Rn	C8..CF	4	160ns	12	480ns	3
XCH A, direct	C5	8	320ns	12	480ns	1.5
XCH A, @Ri	C6..C7	4	160ns	12	480ns	3
XCHD A, @Ri	D6..D7	4	160ns	12	480ns	3
CLR C	C3	4	160ns	12	480ns	3
CLR bit	C2	8	320ns	12	480ns	1.5
SETB C	D3	4	160ns	12	480ns	3
SETB bit	D2	8	320ns	12	480ns	1.5
CPL C	B3	4	160ns	12	480ns	3
CPL bit	B2	8	320ns	12	480ns	1.5
ANL C, bit	82	8	320ns	24	960ns	3
ANL C, /bit	B0	8	320ns	24	960ns	3
ORL C, bit	2	8	320ns	24	960ns	3
ORL C, /bit	A0	8	320ns	24	960ns	3
MOV C, bit	A2	8	320ns	12	480ns	1.5
MOV bit, C	92	8	320ns	24	960ns	3
ACALL addr 11	Hex code					
Hex codes=11, 31, 51, 71, 91, B1, D1, or F1	Byte 1	12	480ns	24	960ns	2
LCALL addr 16	12	16	640ns	24	960ns	1.5
RET	22	16	640ns	24	960ns	1.5
RETI	32	16	640ns	24	960ns	1.5
AJMP addr 11	Hex code					
Hex code=01, 21, 41, 61, 81, A1, C1, or E1	Byte 1	12	480ns	24	960ns	2
LJMP addr 16	2	16	480ns	24	960ns	1.5
JMP @A+DPTR	73	12	480ns	24	960ns	2
SJMP rel	80	12	480ns	24	960ns	2
JZ rel	60	12	480ns	24	960ns	2
JNZ rel	70	12	480ns	24	960ns	2
JC rel	40	12	480ns	24	960ns	2
JNC rel	50	12	480ns	24	960ns	2
JB bit, rel	20	16	640ns	24	960ns	1.5
JNB bit, rel	30	16	640ns	24	960ns	1.5
JBC bit, rel	10	16	640ns	24	960ns	1.5
CJNE A, direct, rel	B5	16	640ns	24	960ns	1.5
CJNE A, #data, rel	B4	16	640ns	24	960ns	1.5
CJNE Rn, #data, rel	B8..BF	16	640ns	24	960ns	1.5
CJNE @Ri, #data, rel	B6..B7	16	640ns	24	960ns	1.5
DJNZ Rn, rel	D8..DF	12	480ns	24	960ns	2
DJNZ direct, rel	D5	16	640ns	24	960ns	1.5
NOP	00	4	160ns	12	480ns	3

### INSTRUCTION SPEED SUMMARY Table 5-2

INSTRUCTION CATEGORY	
Total Instructions:	One Cycle One Byte
Total Instructions:	Two Cycle One Byte
Total Instructions:	Two Cycle Two Bytes X1.5
Total Instructions:	Two cycle Two Bytes X3.0
Total Instructions:	Three Cycle One Byte
Total Instructions:	Three Cycle Two Bytes
Total Instructions:	Three Cycle Three Bytes
Total Instructions:	Four Cycle One Byte
Total Instructions:	Four Cycle Three Bytes
Total Instructions:	Five Cycle One Byte
Average Across all Instructions	
Total Opcodes:	One Cycle One Byte
Total Opcodes:	Two Cycle One Byte
Total Opcodes:	Two Cycle Two Bytes X1.5
Total Opcodes:	Two Cycle Two Bytes X3.0
Total Opcodes:	Three Cycle One Byte
Total Opcodes:	Three Cycle Two Bytes
Total Opcodes:	Three Cycle Three Bytes
Total Opcodes:	Four Cycle One Byte
Total Opcodes:	Four Cycle Three Bytes
Total Opcodes:	Five Cycle One Byte
Average Across all Instructions	

### Average Across all Instructions

QUANTITY	SPEED ADVANTAGE
37	3.0
4	3.0
27	1.5
11	3.0
4	2.0
8	2.0
7	2.0
2	1.5
9	1.5
2	2.4
111	2.3
126	3.0
6	3.0
35	1.5
27	3.0
4	2.0
29	2.0
7	2.0
2	1.5
17	1.5
2	2.4
255	2.5

## SECTION 6: MEMORY ACCESS

The High-Speed Microcontroller follows the memory interface convention established for the industry standard 80C51/80C31. Products in the family may vary, so refer to the specific product data sheet for any potential differences. Like the 8051 series, the High-Speed Microcontroller uses two memory segments. These are program memory and data memory. Program memory is read-only and is usually implemented in ROM or EPROM. Data memory is read/write and is commonly implemented in SRAM.

Memory areas can be implemented as either on-chip, off-chip, or a combination. When using devices without internal program memory, or if the maximum address of on-chip program or data memory is exceeded, the device will perform an external memory access using the Expanded memory bus on ports 0 and 2. While serving as a memory bus, port 0 and port 2 do not function as I/O ports, following the standard 8051 convention of addressing external memory. The PSEN signal will go active low to serve as a chip enable or output enable when performing a code fetch from external memory. Products with no on-chip program memory such as the DS80C320 will always use the Expanded bus. These devices have no Port 0 latch since the port is dedicated for memory operations. Devices which incorporate on-chip MOVX data memory operate in a similar fashion, except that the  $\overline{RD}$  and  $\overline{WR}$  signals serve as chip enables when accessing an external SRAM.

Program execution begins at address 0000h. This is the reset vector. Any reset will cause the next program fetch to begin at this location. Subsequent branches and interrupts determine how the memory fetch deviates from sequential addressing. Since all programs begin at 0000h, this will be the beginning address of all program execution. If on-chip program memory is present, program execution will begin at internal location 0000h, otherwise external program memory will be used.

### INTERNAL PROGRAM MEMORY

Some members of the High-Speed Microcontroller family incorporate internal EPROM or ROM for program storage. On-chip program memory begins at address 0000h and is contiguous through the amount of on-chip memory. Exceeding the maximum address of on-chip memory will cause the device to perform an external memory access using the Expanded memory bus on ports 0 and 2. For example, if the on-chip program memory is 16KB, then it lies between 0000h and 3FFFh

in a contiguous area. Thus a fetch at program memory location 4000h would be directed to the Expanded bus. Restricting memory operations within the on-chip memory allows ports 0 and 2 to be used for general purpose I/O. For more information concerning memory size for a specific device, consult the corresponding data sheet.

The High-Speed Microcontroller family was designed to be compatible with industry standard 87C51FB programming tools. A number of third-party device programmers are available which support Dallas Semiconductor products. In addition, Dallas Semiconductor manufactures the DS87000 Microcontroller Programmer, specifically designed for programming EPROM-based members of the High-Speed Microcontroller family.

### INTERNAL DATA MEMORY

Some members of the High-Speed Microcontroller family incorporate internal SRAM for additional data storage. This memory is addressed via MOVX commands, and is in addition to the 256 bytes of scratchpad memory. On-chip data memory begins at address 0000h and is contiguous through the amount of on-chip memory. Exceeding the maximum address of on-chip memory will cause the device to perform an external memory access using the Expanded memory bus on ports 0 and 2. For example, if the on-chip program memory is 1KB, then it lies between 0000h and 03FFh in a contiguous area. Thus a MOVX instruction affecting memory location 0400h would be directed to the Expanded bus.

Another advantage of internal data memory is that it guarantees a 2 machine cycle data memory access. This data can be made nonvolatile on the DS87C530 through the use of an external battery. Restricting memory operations within the on-chip memory allows ports 0 and 2 to be used for general purpose I/O. For more information concerning memory size for a specific device, consult the corresponding data sheet.

Upon a power-on reset, the internal data memory area is disabled and transparent to the system map. Any memory access between 0000h and FFFFh will be directed to the Expanded bus. This allows the device to remain drop-in compatible with existing 87C52 designs. To enable the internal SRAM area, software must configure the Data Memory Enable bits DME1, DME0 (PMR.1-0). The three memory configurations shown in

Table 6–1 are supported to allow either external data memory access via the expanded bus, internal data memory access, or read-only access to the EPROM

System Control Byte. Note that these bits are cleared after a reset, so access to the internal data memory is prohibited until these bits are modified.

**DATA MEMORY ACCESS CONTROL** Table 6–1

DME1	DME0	DATA MEMORY ADDRESS RANGE	DATA MEMORY LOCATION
0	0	0000h–FFFFh	External Data Memory (default)
0	1	0000h–03FFh 0400h–FFFFh	Internal Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h–03FFh 0400h–FFFBh FFFC FFFDh–FFFFh	Internal Data Memory Reserved System Control Byte (Read only) Reserved

### ROMSIZE FEATURE

Members of the High-Speed Microcontroller family which incorporate internal program memory allow the system to dynamically vary the on-chip memory size. This permits the device to reconfigure the upper limit of on-chip memory, allowing a portion of the memory to be mapped off-chip. The size of on-chip memory can vary from 0KB to the full range of memory, allowing the device to behave like a device with less on-chip memory.

This feature has two primary uses. In the first instance, it allows the device to act as a bootstrap loader for a Flash memory or nonvolatile SRAM (NVS RAM). The internal program memory can contain a bootstrap loader, which can program the external memory device. Secondly, this method can be used to increase the amount of available program memory from 64KB to 80KB without bank-switching.

The maximum amount of on-chip memory is selected by configuring the ROM Size Select register bits RMS2, RMS1, RMS0 (ROMSIZE.2–0). The modification of the ROMSIZE register must be followed by a 2 machine cycle delay, such as executing two NOP instructions, before jumping to the new address range. Interrupts must be disabled during this operation, because a jump to the interrupt vector during the changing of the memory map can cause erratic results. In addition, modification of the ROMSIZE register must be done from a location that will be valid both before and after the on-chip memory configuration. If off-chip memory access is planned, it is recommended that ports 0 and 2 not be used as general purpose I/O, as their state will be disturbed by the memory operations. The settings for

the ROM Size Select register are shown in Table 6–2. Note that the memory configurations shown are not available on all devices.

**ROMSIZE REGISTER SETTINGS** Table 6–2

RMS2	RMS1	RMS0	Max. On-chip ROM
0	0	0	0KB
0	0	1	1KB
0	1	0	2KB
0	1	1	4KB
1	0	0	8KB
1	0	1	16KB
1	1	0	32KB
1	1	1	64KB

After reset, a device with internal program memory will reset the ROMSIZE bits to their default setting. This will be the maximum amount of on-chip memory for that device. The procedure to reconfigure the amount of on-chip memory is as follows:

1. Jump to a location in program memory that will be unaffected by the change,
2. Disable interrupts by clearing the EA bit (IE.7),
3. Write AAh to the Timed Access Register (TA;C7h),
4. Write 55h to the Timed Access Register (TA;C7h),
5. Modify the ROM Size Select bits (RMS2–RMS0),



6. Delay 2 machine cycles (2 NOP instructions),
7. Enable interrupts by setting the EA bit (IE.7).

If the 0KB of internal program memory setting is selected, extra precautions must be taken. In this case, it will be necessary to duplicate the interrupt vector table in external program memory. This is because the interrupt vector table is located in the lower 1KB of memory, and the device will automatically redirect any fetches from the interrupt vector table to external memory. Care must be exercised when assembling or compiling the program so that all the modules are located at the correct starting address, including the interrupt vector table.

### PROGRAM MEMORY INTERCONNECT

The program memory interconnect scheme for the High-Speed Microcontroller family is shown in Figure 6-1. This example uses the DS80C320 and one 32K x 8 EPROM. The Program Store Enable (PSEN) signal is used to provide an output enable to the EPROM. It can also be used to provide a chip enable, but this produces less favorable timing. The address LSB and data are multiplexed on port 0, and the address MSB is provided on port 2. An external latch, shown in the diagram as a 74F373, is used to latch the lower byte of the address to the memory device. The Address Latch Enable (ALE) signal controls the timing of the latch so that the operation is performed in the proper sequence. The signals and relative timing for a program access are shown in Figure 6-2.

When implementing a high speed memory interface, the F series (or faster) logic should be used. HC logic will have worst case propagation delays that are too long. Specifications for all devices should be checked. More information on memory interface timing can be found in Application Note 57, DS80C320 Memory Interface Timing, and Application Note 85, High Speed Microcontroller Interface Timing.

The first product in the family, the DS80C320, provides an extremely high speed interface to external ROM or EPROM. This assures that the user can use the slow-

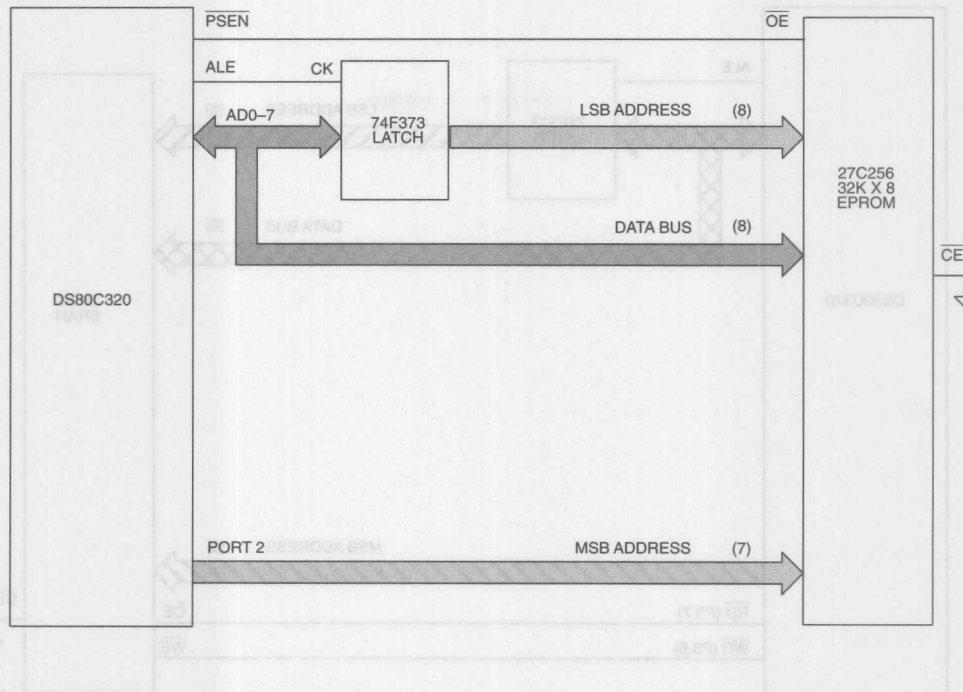
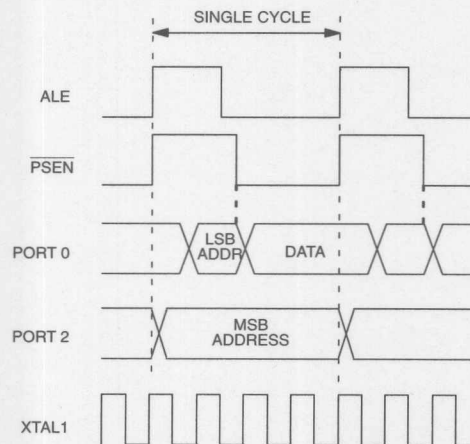
est, and least expensive, memory device for a given crystal speed. The DS80C320 provides very fast slew rates, but controls ringing and overshoot. Fast slew rates allow the maximum possible time for memory access. In most cases, however, these aspects will be transparent to the user. Refer to the electrical specifications for exact timing of each product.

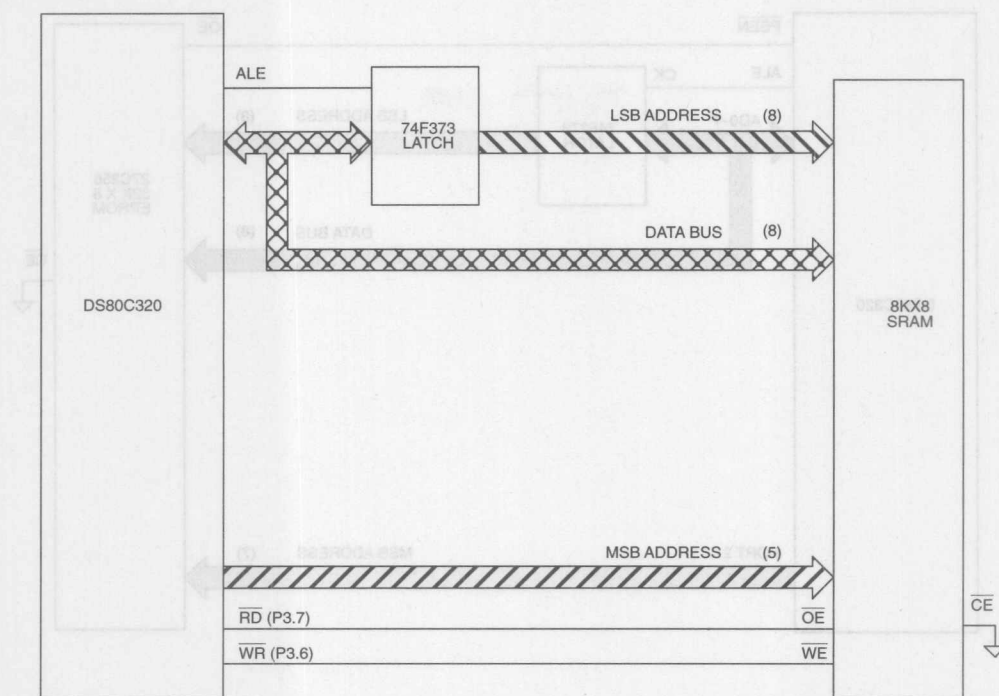
### DATA MEMORY INTERCONNECT

As described in Section 4, the High-Speed Microcontroller provides a small amount of RAM mapped as registers for on-chip direct access. This is not considered data memory and does not fall into the memory map. Systems that require more RAM or memory mapped peripherals must use the data memory area. This segment is a 64KB space located between 0000h and FFFFh. It is reached using the MOVX instruction. Any use of this instruction automatically accesses the data area. Although, the original 8051 convention placed all data memory off-chip, many members of the High-Speed Microcontroller family contain some data memory on-chip.

From a software standpoint, the physical location of the data area is not relevant because the same instructions are used. Like the program segment, if software accesses a data address that is above the on-chip data area, this access will automatically be routed to the Expanded bus. Thus data or peripherals that are off-chip can be used in conjunction with on-chip memory by selecting addresses that do not overlap. As an example, if the microcontroller has 1KB of on-chip data memory, then a MOVX instruction at location 0400h will be directed off-chip via the Expanded bus.

The physical connection of off-chip data memory is shown in Figure 6-3. This illustrates a DS80C320 with interfaced with an 8K SRAM. The data memory map begins at address 0000h since the DS80C320 has no on-chip data memory. A similar interconnection scheme would be implemented if a device with internal data memory, such as the DS87C520 would be used. Note that any external memory that overlapped the range of on-chip data memory would not be used.

**PROGRAM MEMORY INTERFACE** Figure 6-1**PROGRAM MEMORY SIGNALS** Figure 6-2

**DATA MEMORY INTERFACE** Figure 6-3

## DATA MEMORY ACCESS

As mentioned above, the High-Speed Microcontroller uses the MOVX instruction for data memory access. This includes off-chip RAM and memory mapped peripherals needing read/write access. Several aspects of the MOVX operation have been enhanced as compared to the original 8051. The principal improvements are in the areas of the MOVX timing and the Data Pointer.

The MOVX instruction is used to generate read/write access to off-chip address locations. It has several addressing modes. The MOVX can be used to reach a 256 byte block by using the MOVX @ Ri command. This instruction uses the value in the designated working register to address one of 256 locations. The upper byte of the address is supplied by the value in the Port 2 latch. A second way to access data is the Data Pointer (DPTR). This 16-bit register provides an absolute address for data memory access. 16-bits cover the entire 64KB area. Thus the DPTR serves as a pointer to memory. Using the DPTR, the relevant instruction is MOVX @DPTR.

The original 8051 contained one DPTR. While this provides access to the entire memory area, it is difficult to move data from one address to another. The High-Speed Microcontroller provides two Data Pointers. Thus software can load both a source and a destination address. The MOVX instruction will use the active pointer to direct the off-chip address. The active pointer is selected by the Data Pointer Select (DPS).

The Data Pointers are called DPTR0 and DPTR1. DPTR0 is located at SFR addresses 82h and 83h. These are the locations used by the original 8051. No modification of standard code is needed to use DPTR0. The new DPTR is located at SFR 84h and 85h. The Data Pointer Select bit (DPS) chooses the active pointer and is located at the LSb of the SFR location 86h. No other

bits in register 86h have any effect and are set to 0. When DPS is set to 0, the DPTR0 is active. When set to 1, DPTR1 is used.

The user switches between data pointers by toggling the DPS bit (LSb of register 86h). The INC instruction is the fastest way to accomplish this. All DPTR-related instructions use the currently selected DPTR for any activity. Therefore only one instruction is required to switch from a source to a destination address. Using the Dual Data Pointer saves code from needing to save source and destination addresses when doing a block move. Once loaded, the software simply switches between DPTR0 and DPTR1. Sample code listed below illustrates the saving from using the dual DPTR. The relevant register locations are summarized as follows.

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (LSb)

The example program listed below was original code written for an 8051 and requires a total of 1869 machine cycles on the DS80C320. This takes 299  $\mu$ s to execute at 25 MHz. The new code using the Dual DPTR requires only 1097 machine cycles taking 175.5  $\mu$ s. The Dual DPTR saves 772 machine cycles or 123.5  $\mu$ s for a 64 byte block move. Since each pass through the loop saves 12 machine cycles when compared to the single DPTR approach, larger blocks gain more efficiency using this feature.

A typical application of the Dual Data Pointer is moving data from an external RAM to a memory mapped display. Another application would be to retrieve data from a stored table, process it using a software algorithm, then store the result in a new table.

**64 BYTE BLOCK MOVE WITH DUAL DATA POINTER**

; SH and SL are high and low byte source address.  
 ; DH and DL are high and low byte of destination address.  
 ; DPS is the data pointer select. Reset condition is DPS=0, DPTR0 is selected.

			# CYCLES
DPS	EQU 86h	; TELL ASSEMBLER ABOUT DPS	
MOV	R5, #64	; NUMBER OF BYTES TO MOVE	2
MOV	DPTR, #DHDL	; LOAD DESTINATION ADDRESS	3
INC	DPS	; CHANGE ACTIVE DPTR	2
MOV	DPTR, #SHSL	; LOAD SOURCE ADDRESS	2

MOVE:

; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64

MOVX	A, @DPTR	; READ SOURCE DATA BYTE	2
INC	DPS	; CHANGE DPTR TO DESTINATION	2
MOVX	@DPTR, A	; WRITE DATA TO DESTINATION	2
INC	DPTR	; NEXT DESTINATION ADDRESS	3
INC	DPS	; CHANGE DATA POINTER TO SOURCE	2
INC	DPTR	; NEXT SOURCE ADDRESS	3
DJNZ	R5, MOVE	; FINISHED WITH TABLE?	3

**64 BYTE BLOCK MOVE WITHOUT DUAL DATA POINTER**

; SH and SL are high and low byte source address.  
 ; DH and DL are high and low byte of destination address.

			# CYCLES
MOV	R5, #64d	; NUMBER OF BYTES TO MOVE	2
MOV	DPTR, #SHSL	; LOAD SOURCE ADDRESS	3
MOV	R1, #SL	; SAVE LOW BYTE OF SOURCE	2
MOV	R2, #SH	; SAVE HIGH BYTE OF SOURCE	2
MOV	R3, #DL	; SAVE LOW BYTE OF DESTINATION	2
MOV	R4, #DH	; SAVE HIGH BYTE OF DESTINATION	2

MOVE:

; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64

MOVX	A, @DPTR	; READ SOURCE DATA BYTE	2
MOV	R1, DPL	; SAVE NEW SOURCE POINTER	2
MOV	R2, DPH	;	2
MOV	DPL, R3	; LOAD NEW DESTINATION	2
MOV	DPH, R4	;	2
MOVX	@DPTR, A	; WRITE DATA TO DESTINATION	2
INC	DPTR	; NEXT DESTINATION ADDRESS	3
MOV	R3, DPL	; SAVE NEW DESTINATION POINTER	2
MOV	R4, DPH	;	2
MOV	DPL, R1	; GET NEW SOURCE POINTER	2
MOV	DPH, R2	;	2
INC	DPTR	; NEXT SOURCE ADDRESS	3
DJNZ	R5, MOVE	; FINISHED WITH TABLE?	3



## DATA MEMORY TIMING

Data memory timing refers to the execution of the MOVX instruction. This instruction includes a program fetch memory access, then a read or write memory access. The program fetch for a MOVX instruction is no different from any other instruction. The unique timing occurs for the second memory operation when data is accessed.

As described in Section 5, the High-Speed Microcontroller uses four oscillator clocks for each machine cycle. A machine cycle involves one memory access. Generally, an instruction using two memory accesses would be a two machine cycle instruction (except for branches, MUL, DIV, INC DPTR, MOVC, and MOVX). The MOVX instruction is unique in that the user determines the time allowed for a data memory access. This feature is called the Stretch MOVX instruction.

The High-Speed Microcontroller allows the application software to adjust the speed of data memory access. The microcontroller is capable of performing the MOVX in as little as two machine cycles. Since one machine cycle is used for the program fetch, this leaves one machine cycle to perform the actual data memory access. However, this value can be adjusted as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic. Even in high-speed systems, it may not be necessary to perform data memory access at full speed. In addition, there are a variety of slower memory mapped peripherals such as LCD displays or UARTs.

When using a MOVX instruction, the user controls the time for which a read or write strobe is kept active. Setup and hold times are also adjusted. Thus the Stretch value will be selected to provide a long enough memory strobe to satisfy the access time of the target device.

The Stretch MOVX is controlled by a value in a special function register described below. This allows the user to select a stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. This leaves one machine cycle to actually read or write data. A Stretch of seven will result in a MOVX of nine cycles. The time is added to the middle of the memory strobe, creating a very long read or write cycle. The

Stretch value can be changed dynamically under software control depending on the type of memory or peripheral to be accessed.

On reset, the Stretch value will default to a one, resulting in a three cycle MOVX. Therefore, data memory access will not be performed at full speed. This is a convenience to existing designs that may not have fast RAM in place. When maximum speed is desired, the software should select a Stretch value of zero. Note that faster RAMs will be needed. When using very slow RAM or peripherals, a larger stretch value can be selected. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven results in a wider read/write strobe allowing more time for memory/peripherals to respond. The microcontroller stretches the read/write strobe and all related timing. The full speed access is shown in Figure 6-4. Note that this is not the reset default case. A three cycle MOVX is shown in Figure 6-5A. This is the reset default condition. To modify the MOVX timing, the Stretch value in the Clock Control register described below must be changed. Figure 6-5B shows the timing for a four cycle MOVX (Stretch=2).

Table 6-1 below shows the resulting strobe widths for each Stretch value. The memory stretch is implemented using the Clock Control Special Function Register at SFR location 8Eh. The stretch value is selected using bits CKCON.2-0. In the table, these bits are referred to as M2 through M0. Note that the Stretch time can be dynamically varied, allowing fast RAMs but slow peripherals. The first stretch allows the use of common 120 ns or 150 ns RAMs without dramatically lengthening the memory access.

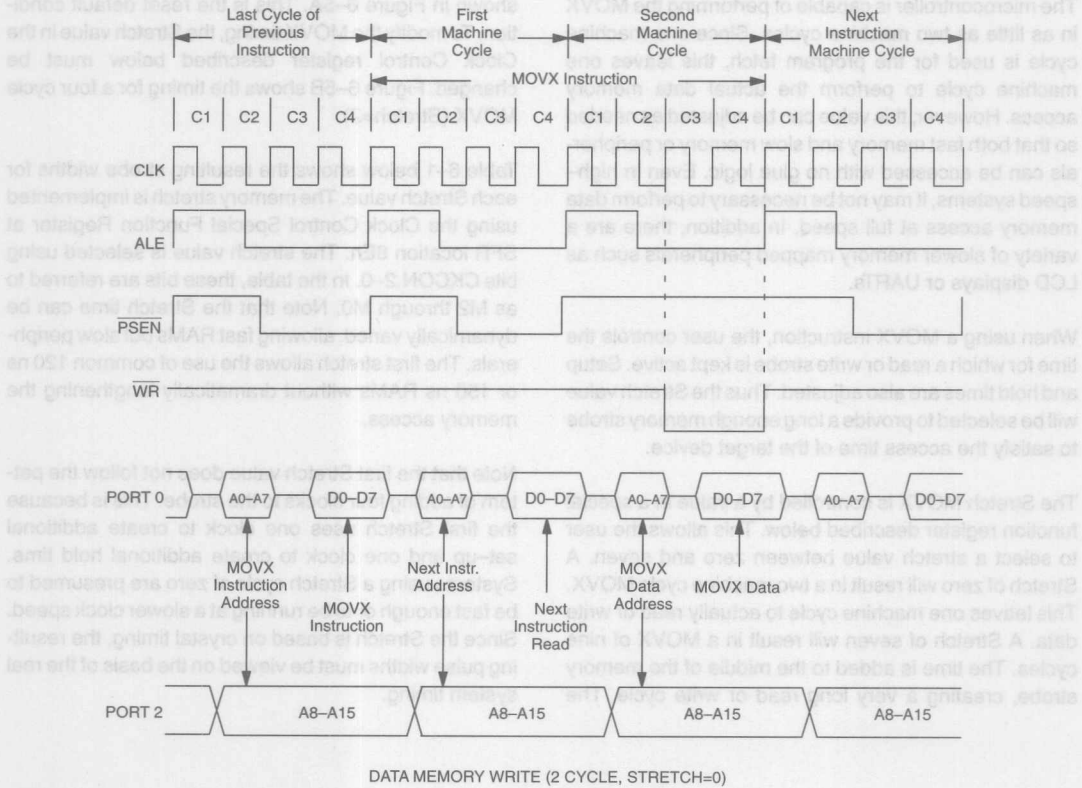
Note that the first Stretch value does not follow the pattern of adding four clocks to the strobe. This is because the first Stretch uses one clock to create additional set-up and one clock to create additional hold time. Systems using a Stretch cycle of zero are presumed to be fast enough or to be running at a slower clock speed. Since the Stretch is based on crystal timing, the resulting pulse widths must be viewed on the basis of the real system timing.

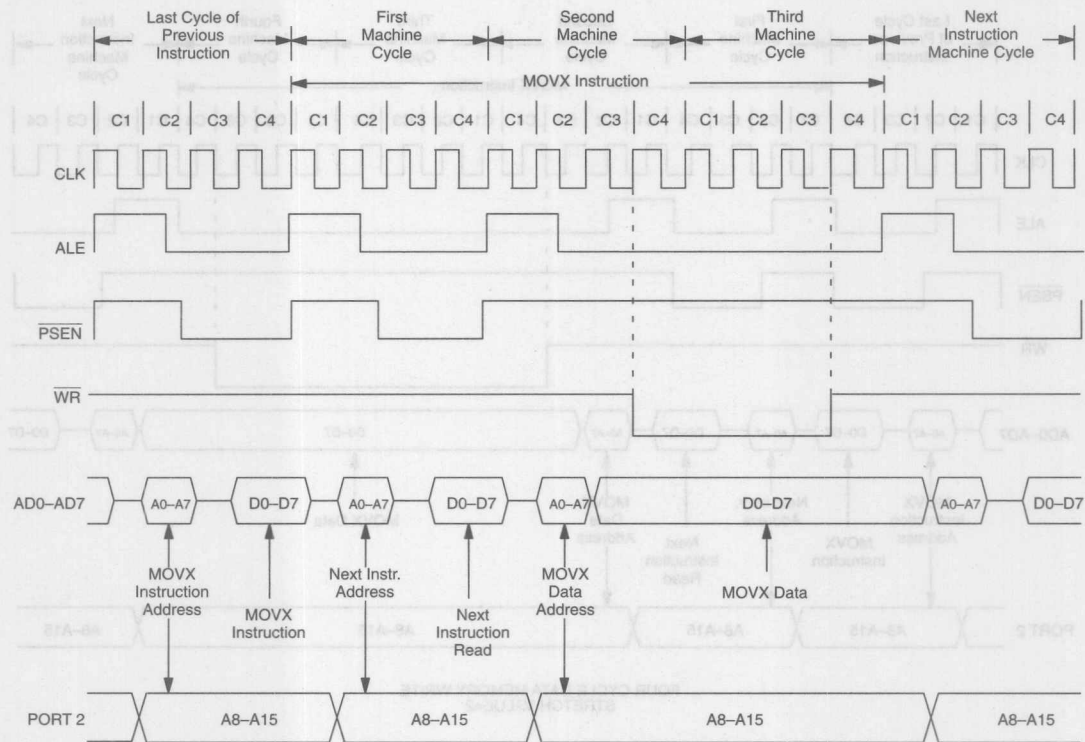
DATA MEMORY CYCLE STRETCH VALUES Table 6-1

CKCON.2-0			MEMORY CYCLES	RD OR WR STROBE WIDTH		
M2	M1	M0		IN CLOCKS	t @ 25 MHz	t @ 12 MHz
0	0	0	2	2	80 ns	167 ns
0	0	1	3 (default)	4	160 ns	333 ns
0	1	0	4	8	320 ns	667 ns
0	1	1	5	12	480 ns	1000 ns
1	0	0	6	16	640 ns	1333 ns
1	0	1	7	20	800 ns	1667 ns
1	1	0	8	24	960 ns	2000 ns
1	1	1	9	28	1120 ns	2333 ns

Note: These numbers represent nominal values. Actual timing may vary slightly.

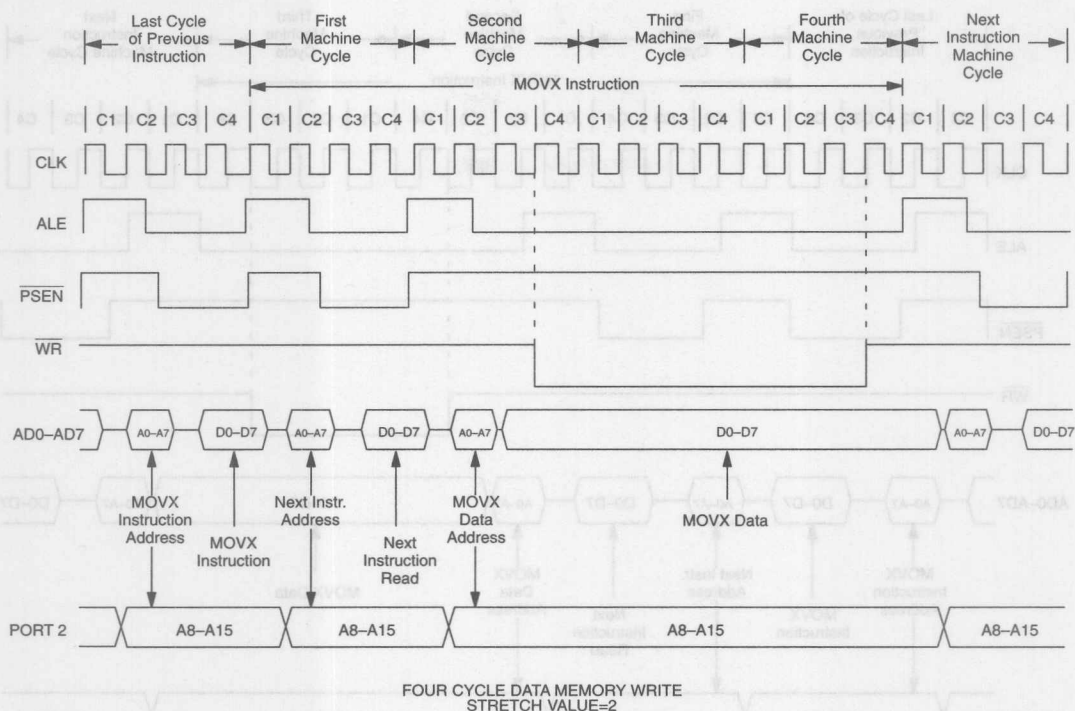
FULL SPEED MOVX INSTRUCTION Figure 6-4



**THREE CYCLE MOVX INSTRUCTION** Figure 6-5a

THREE CYCLE DATA MEMORY WRITE (RESET DEFAULT)  
STRETCH VALUE=1

# **FOUR CYCLE MOVX INSTRUCTION Figure 6-5b**



## SECTION 7: POWER MANAGEMENT

The High-Speed Microcontroller has several features that relate to power consumption and management. They provide a combination of controlled operation in unreliable power applications and reduced power consumption in portable or battery powered applications. The range of features is shown below with details to follow.

### POWER MANAGEMENT

EARLY WARNING POWER FAIL INTERRUPT  
POWER FAIL/POWER ON RESET  
BAND-GAP SELECT  
WATCHDOG WAKE UP FROM IDLE

### POWER SAVING

IDLE MODE  
STOP MODE  
RING WAKE UP FROM STOP  
POWER MANAGEMENT MODES

## PRECISION VOLTAGE MONITOR

The High-Speed Microcontroller uses a precision band-gap reference and other analog circuits to monitor the state of the power supply during power-up and power-down transitions. Other microcontroller systems would require external circuits to perform these functions. The band-gap reference provides a precise voltage to compare with  $V_{CC}$ . When  $V_{CC}$  begins to drop, the Power Monitor compares it to its reference. This enables the analog circuits to detect when  $V_{CC}$  passes through predetermined thresholds,  $V_{PFW}$  and  $V_{RST}$ . These are specified in the individual product data sheets.

## EARLY WARNING POWER FAIL INTERRUPT

Devices which incorporate the precision voltage reference have the ability to generate a power-fail interrupt and/or reset in response to a low supply voltage. When  $V_{CC}$  reaches the  $V_{PFW}$  threshold, the microcontroller can generate an power-fail Interrupt. This early warning of supply voltage failure allows the system time to save critical parameters in nonvolatile memory and put external functions in a safe state.

The power-fail interrupt is optional and is enabled using the Enable Power Fail Warning Interrupt (EPFI) bit at WDCON.5. If enabled,  $V_{CC}$  dropping below  $V_{PFW}$  will cause the device to vector to address 33h. The Power-fail Interrupt status bit, PFI (WDCON.4), will be set any

time  $V_{CC}$  transitions below  $V_{PFW}$ . This flag is not cleared when  $V_{CC}$  is above  $V_{PFW}$ , and software should clear it immediately after reading it. As long as the condition exists, PFI will be immediately set again by hardware.

A typical application of the PFI is to place the device into a "safe mode" when a power loss appears imminent. When the interrupt occurs, the code vectors to location 33h. At this time, software can disable the interrupt, save any critical data, clear PFI, then continually poll the status of the power supply via the PFI flag. As long as PFI is set, power is still below  $V_{PFW}$ . If power returns to the proper level, PFI will not be set once cleared by software. This indicates a safe operating condition. If power continues to fall, a power-fail reset will be invoked automatically.

## POWER-FAIL RESET

Devices which incorporate the power-fail reset will automatically invoke a reset when  $V_{CC}$  drops below  $V_{RST}$ . This will halt device operation, and place all outputs in their reset state. This state will continue to be held until  $V_{CC}$  drops below the voltage necessary to power the port pins. Because  $V_{RST}$  is lower than  $V_{PFW}$ , the microcontroller has the option to use the power-fail interrupt to place the device into a "safe" state before the device halts operation with a power-fail reset. This feature is automatic on devices which incorporate the power-fail reset feature, and cannot be disabled.

## POWER ON RESET

When  $V_{CC}$  is applied to a system using the High-Speed Microcontroller, the device will hold itself in reset until power is within tolerance and stable. An internal band-gap reference provides a highly accurate and stable means of detecting power supply levels. It requires no external circuits to accomplish this. As power rises, the processor will stay in a reset state until  $V_{CC} > V_{RST}$ . As  $V_{CC}$  rises above  $V_{RST}$ , internal analog circuits will detect this and activate the on-chip crystal oscillator. On-chip hardware will then count 65536 oscillator clocks. During this count,  $V_{CC}$  must remain above  $V_{RST}$  or the process restarts. If an off-chip clock source is used, then clock counting still begins once  $V_{CC} > V_{RST}$ . This count period is used to make certain that power is within tolerance, and that the oscillator has time to stabilize. This provides a very controlled and predictable start-up condition.



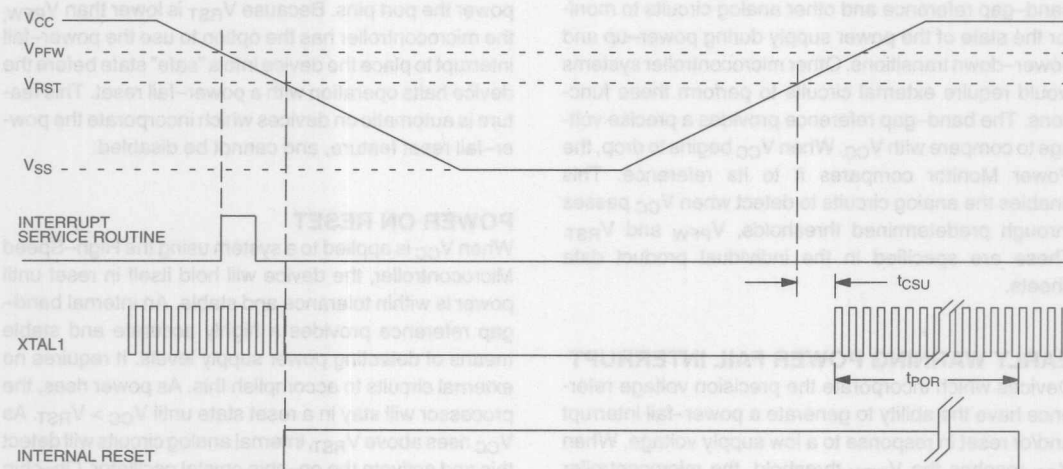
Once the 65536 count period has elapsed, the reset condition is removed automatically, and software execution will begin at the reset vector location of 0000h. Software will be able to detect the power on reset condition using the Power-On Reset (POR) flag. POR is located at WDCON.6. This bit will be high to indicate that a Power-on Reset has occurred. It should then be cleared by software.

The complete power cycle operation is shown in Figure 7-1. Note that the interrupt threshold is fixed, but the interrupt itself is optional. Reset thresholds are also fixed and the reset operation is transparent. It requires no external components and no action by software to control reset operation.

### BAND-GAP SELECT

When present, the band-gap reference will provide a precise voltage reference for the power-fail monitor circuitry. The band-gap is normally disabled automatically

### POWER CYCLE OPERATION Figure 7-1



upon entering Stop mode to provide the lowest power state. Since the band-gap is inactive, there can be no power-fail interrupt and no power-fail reset, similar to a traditional 8051.

If the use of the power-fail features are desired in Stop mode, the BGS bit (EXIF, 91h) may be used. When set to a logic 1 by software, the band-gap reference and associated power monitor circuits will remain active in Stop mode. The price of this feature is higher power supply current requirements. In Stop mode with the band-gap reference disabled (default), the processor draws approximately 1  $\mu$ A. With the band-gap enabled, it draws approximately 50  $\mu$ A.

BGS allows the user to decide whether the control circuitry and its associated power consumption are needed. If the application is such that power will not fail while in Stop or if it does not matter that power fails, the BGS should be set to 0 (default). If power can fail at any time and cause problems, the BGS should be set to 1.

### WATCHDOG WAKE UP

The Watchdog Wake up is more of an application than a feature. It allows a system to enter the Idle mode for power savings, then to wake up periodically to sample the external world. Idle mode is a low power state described below. Any of the programmable timers can perform this function, but the Watchdog allows a much longer period to be selected. At 12 MHz, the maximum Watchdog time-out is over 5.5 seconds. This contrasts with 0.78 seconds using the 16-bit timers. Software that uses the Watchdog as a wake up alarm should only enable the Watchdog Interrupt and not the Reset. Note that the Watchdog cannot be used to wake the system while in Stop mode since no clocks are running. Stop mode is described below.

### POWER MANAGEMENT SUMMARY

The following is a summary of the power management bits and those that are useful or related. They are contained in the register locations WDCON;D8h, EIE;E8h, EXIF;91h, and PCON;87h.

**WDCON.6** POR – Power on Reset. Hardware will set this bit on a power up condition. Software can read it, but must clear it manually. This bit assists software in determining the cause of a reset.

**WDCON.5** EPFI – Enable Power-fail Interrupt. Setting this bit to 1 enables the Power-fail interrupt. This will occur when  $V_{CC}$  drops to approximately 4.5 volts, and the processor vectors to location 33h. Setting this bit to a 0 turns off the Power-fail Interrupt.

**WDCON.4** PFI – Power Fail Interrupt Flag. Hardware will set this bit to a 1 when a Power-fail condition occurs. Software must clear the bit manually. Writing a 1 to this bit will force an interrupt, if enabled.

**WDCON.3** WDIF – Watchdog Interrupt Flag. If the Watchdog Interrupt is enabled (EIE.4), hardware will set this bit to indicate that the Watchdog Interrupt has occurred. If the interrupt is not enabled, this bit indicates that the time-out has passed. If the Watchdog Reset is enabled (WDCON.1), the user has 512 clocks to strobe the Watchdog prior to a reset. Software or any reset can clear this flag.

**WDCON.2** WTRF – Watchdog Timer Reset Flag. Hardware will set this bit when the Watchdog Timer causes a reset. Software can read it, but must clear it manually. A Power-fail Reset will also clear the bit. This bit assists software in determining the cause of a reset. If EWT=0, the Watchdog Timer will have no effect on this bit.

**WDCON.1** EWT – Enable Watchdog Timer Reset. Setting this bit will turn on the Watchdog Timer Reset function. The Interrupt will not occur unless the EWDI bit in the EIE register is set. A reset will occur according to the WD1 and WD0 bits in the CKCON register. Setting this bit to a 0 will disable the reset but leave the timer running.

**WDCON.0** RWT – Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time-out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the time-out has elapsed. There is no need to set the RWT bit to a 0 because it is self-clearing.

**EIE.4** EWDI – Enable Watchdog Interrupt. Setting this bit in software enables the Watchdog Interrupt.

**EXIF.0** BGS – Band-Gap Select. Setting this bit to a 1 will allow the use of the Band-gap voltage reference while in Stop mode. Since this function uses as much as 50  $\mu$ A, the band-gap is optional in Stop mode. Setting this bit to a 0 will turn off the Band-gap while in Stop mode. When BGS=0, no Power-fail interrupt or Power-fail Reset will be available in Stop mode.

**PCON.1** STOP. When this bit is set, the program stops execution, clocks are stopped, and the CPU enters power-down mode.

**PCON.0** IDLE. Program execution halts leaving timers, serial ports, and clocks running.

**EXIF.2** RGMD – Ring Oscillator Mode. Hardware will set this status bit to a 1 when the clock source is the Ring Oscillator. Hardware will set this status bit to a 0 when the crystal is the clock source. Refer to RGSL below for operation of the Ring Oscillator.

**EXIF.1** RGSL – Ring Oscillator Select. When set to a 1 by software, the High-Speed Microcontroller will use a Ring Oscillator to come out of Stop mode without waiting for crystal start-up. This allows an instantaneous start-up when coming out of Stop mode. It is useful if software needs to perform a short task, then return to Stop. It is also useful if software must respond quickly to an external event. After the crystal has performed 65,536 cycles, hardware will switch to the crystal as its clock source. The RGMD status bit reports on this changeover. When RGSL is set to a 0, the High-Speed Microcontroller will delay software execution until after the 65,536 clock crystal startup time. RGSL is only cleared by a power-on reset and is not altered by other forms of reset.

## POWER SAVING

The High-Speed Microcontroller is implemented using full CMOS circuitry for low power operation. It is fully static so the clock speed can be run down to DC. Like other CMOS, the power consumption is also a function of operating frequency. Although the High-Speed Microcontroller is designed for maximum performance, it also provides improved power versus work relationships compared with standard 8051 devices. These topics are discussed in detail below.

The High-Speed Microcontroller provides two modes (other than operating) that allow power conservation. They are similar, but have different merits and drawbacks. These modes are Idle and Stop. In the original 8051, the Stop mode is called Power Down. These modes are invoked in the same manner as the original 8051 series.

## IDLE MODE

Idle mode suspends all CPU processing by holding the program counter in a static state. No program values are fetched and no processing occurs. This saves considerable power versus full operation. The virtue of Idle mode is that it uses half the power of the operating state, yet reacts instantly to any interrupt conditions. All clocks remain active so the timers, Watchdog, Serial Port, and Power Monitor functions are all working. Since all clocks are running, the CPU can exit the Idle state using any of the interrupt sources.

Software can invoke the Idle mode by setting the IDLE bit in the PCON register at location 87h. The bit is

located at PCON.0. The instruction that executes this step will be the last instruction prior to freezing the program counter. Once in Idle, all resources are preserved. There are two ways to exit the Idle mode. First, any interrupt (that is enabled) will cause an exit. This will result in a jump to the appropriate interrupt vector. The IDLE bit in the PCON register will be cleared automatically. On returning from this vector using the RETI instruction, the next address will be the one immediately after the instruction that invoked the Idle state.

The Idle mode can also be removed using a reset. Any of the three reset sources can do this. On receiving the reset stimulus, the CPU will be placed in a reset state and the Idle condition cleared. When the reset stimulus is removed, software will begin execution as for any reset. Since all clocks are active, there will be no delay after the reset stimulus is removed. Note that if enabled, the Watchdog Timer continues to run during Idle and must be supported.

## STOP MODE

Stop mode is the lowest power state that the High-Speed Microcontroller can enter. This is achieved by stopping all on-chip clocks, resulting in a fully static condition. No processing is possible, timers are stopped, and no serial communication is possible. Processor operation will halt on the instruction that sets the STOP bit. The internal amplifier that excites the external crystal will be disabled, halting crystal oscillation in Stop mode. Table 7-1 shows the state of the processor pins in Idle and Stop modes.

Stop mode can be exited in two ways. First, like the 8052 microcontrollers, a non-clocked interrupt such as the external interrupts or the power-fail interrupt can be used. Clocked interrupts such as the watchdog timer, internal timers, and serial ports will not operate in Stop mode. Note that the band-gap reference must be enabled in order to use the power-fail interrupt to exit Stop mode, which will increase Stop mode current. Processor operation will resume with the fetching of the interrupt vector associated with the interrupt that caused the exit from Stop mode. When the interrupt service routine is complete, an RETI will return the program to the instruction immediately following the one that invoked the Stop mode.

A second method of exiting Stop mode is with a reset. The watchdog timer reset is not available as a reset source because no timers are running in Stop mode. An external reset via the RST pin will unconditionally exit the device from Stop mode. If the BGS bit is set, the device will provide a reset while in Stop mode if  $V_{CC}$  should drop below the  $V_{RST}$  level. If the BGS bit is 0, then a dip in power below  $V_{RST}$  will not cause a reset. For example, if  $V_{CC}$  should drop to a level of  $V_{RST}-0.5V$ , then return to the full level, no reset will be generated. For this reason, use of the band-gap reference is recommended if a brownout condition is possible in Stop mode. If power fails completely ( $V_{CC}=0V$ ), then a power on reset will still be performed when  $V_{CC}$  is reapplied regardless of the state of the BGS bit. Processor operation will resume execution from address 0000h like any other reset.

## CRYSTAL RESUME FROM STOP MODE

If the microcontroller does not contain a ring oscillator, or if the RGSL bit is 0, a device exiting Stop mode must restart operation using the external crystal as a clock source. The device will experience a power-on reset delay of 65536 external clock cycles to allow the crystal to begin oscillation and the frequency to stabilize. Once this delay is complete, software will begin execution from either address 0000h or the appropriate interrupt vector, depending on the stimulus to exit Stop mode. The same 65536 external clock cycle delay is performed if an external crystal oscillator is used instead of an external crystal.

**PIN STATES IN POWER SAVING MODES** Table 7-1

DEVICE	MODE	ALE	PSEN	P0 (AD0-7)	P1	P2	P3
DS80C310 DS80C320	Idle or Stop	1	1	Latched <sup>1</sup>	Port data <sup>2</sup>	Latched <sup>3</sup>	Port data <sup>2</sup>
All Others Internal program execution	Idle or Stop	1	1	Port data <sup>2</sup>	Port data <sup>2</sup>	Port data <sup>2</sup>	Port data <sup>2</sup>
All Others External program execution	Idle	1	1	Latched <sup>1</sup>	Port data <sup>2</sup>	Latched <sup>3</sup>	Port data <sup>2</sup>
All Others External program execution	Stop	1	1	Port data <sup>2</sup>	Port data <sup>2</sup>	Port data <sup>4</sup>	Port data <sup>2</sup>

<sup>1</sup> Port exhibits opcode following instruction that sets the Stop bit. Port 0 is operating in true bidirectional mode, and will drive both a logic 1 and a logic 0.

<sup>2</sup> Port reflects data stored in corresponding Port SFR. Port 0 functions as an open-drain output in this mode.

<sup>3</sup> Port exhibits address MSB of opcode following instruction that sets the Stop bit.

<sup>4</sup> Port reflects data stored in corresponding Port SFR. In this mode, the port uses weak pullups. If a bit in the P2 SFR is a 1, the corresponding device pin will transition slowly to a high when the reset state is entered.



**RING OSCILLATOR WAKE UP FROM STOP**

A typical low power application is to keep the processor in Stop mode most of the time. Periodically, the system will wake up (using an external interrupt), take a reading of some condition, then return to sleep. The duration of full power operation is as short as possible. One disadvantage to this method is that the clock must be restarted prior to performing a meaningful operation. This start-up period is a waste of time and power since no work can be performed. The High-Speed Microcontroller provides an alternative.

If the Ring Select (RGSL) is enabled, the High-Speed Microcontroller can exit Stop mode running from an internal Ring Oscillator. Upon receipt of an interrupt, this oscillator can start instantaneously, allowing software execution to begin immediately while the oscillator is stabilizing. Once 65,536 clock cycles have been detected, the CPU will automatically switch to the normal oscillator as its clock source. However, if the required interrupt response is very short, the software can re-enter Stop mode before the crystal is even stable. In this case, Stop mode can be invoked and both oscillators will be stopped.

**SPEED REDUCTION**

The High-Speed Microcontroller is a fully CMOS 8051 compatible microcontroller. It can use significantly less power than other 8051 versions because it is more efficient. As an average, software will run 2.5 times faster on the High-Speed Microcontroller than on other 8051 derivatives. Thus the same job can be accomplished by slowing down the crystal by a factor of 2.5. For example, an existing 8051 design that runs at 12 MHz can run at approximately 4.8 MHz on the High-Speed Microcontroller. At this reduced speed, the High-Speed Microcontroller will have lower power consumption than an 8051, yet perform the same job.

Using the 2.5X factor, Table 7-2 shows the approximate speed at which the High-Speed Microcontroller can accomplish the same work as an 8051. The exact improvement will vary depending on the actual instruction mix. Available crystal speeds must also be considered. Refer to Section 16 for information on instruction timing.

**CRYSTAL VS MIPS COMPARISON** Table 7-2

ORIGINAL 8051 CRYSTAL SPEED	MIPS	HIGH-SPEED MICROCONTROLLER CRYSTAL SPEED
3.57 MHz	0.3	1.4 MHz
7.37 MHz	0.6	2.9 MHz
11.0592 MHz	0.9	4.4 MHz
14.318 MHz	1.2	5.7 MHz
16 MHz	1.3	6.4 MHz
20 MHz	1.6	8 MHz
24 MHz	2.0	9.6 MHz
33 MHz	2.7	13.2 MHz
40 MHz	3.3	16 MHz



## POWER MANAGEMENT MODES

Power consumption in CMOS microcontrollers is a function of operating frequency. The Power Management Mode (PMM) feature, available with some members of the High-Speed Microcontroller family, allows software to dynamically match operating frequency and current consumption with the need for processing power. Instead of the default 4 clocks per machine cycle, power management mode 1 (PMM1) and power management mode 2 (PMM2) utilize 64 and 1024 clocks per cycle respectively to conserve power.

A number of special features have been added to enhance the function of the power management modes. The switchback feature allows the device to almost instantaneously return to divide by 4 mode upon acknowledgment of an external interrupt or a falling edge on a serial port receiver pin. The advantages of this become apparent when one calculates the increased interrupt service time of a device operating in PMM. In addition, a device operating in PMM would normally be

unable to sample an incoming serial transmission to properly receive it. The switchback feature, explained below, allows a device to return to divide by 4 operation in time to receive incoming serial port data and process interrupts with no loss in performance.

The DS87C520 and DS87C530 incorporate a Status register (STATUS;C5h) to prevent the device from accidentally reducing the clock rate during the servicing of an external interrupt or serial port activity. This register can be interrogated to determine if a high priority, low priority, or power fail interrupt is in progress, or if serial port activity is occurring. Based on this information the software can delay or reject a planned change in the clock divider rate.

In addition, the DS87C520 and DS87C530 has the capability to operate from the internal ring oscillator during normal operation, not only during the crystal warm-up period. Table 7-3 summarizes the new control bits associated with the power management features.

STATUS.0	Serial Port 0 Receiver Activity Status	0=Serial port 0 receiver inactive 1=Serial port 0 receiver active	Read Only
STATUS.1	Serial Port 0 Transmitter Activity Status	0=Serial port 0 transmitter inactive 1=Serial port 0 transmitter active	Read Only
STATUS.2	Serial Port 1 Receiver Activity Status	0=Serial port 1 receiver inactive 1=Serial port 1 receiver active	Read Only
STATUS.3	Serial Port 1 Transmitter Activity Status	0=Serial port 1 transmitter inactive 1=Serial port 1 transmitter active	Read Only
STATUS.4	Low Priority Interrupt Status	0=No low priority interrupt in progress 1=Low priority interrupt in progress	Read Only
STATUS.5	High Priority Interrupt Status	0=No high priority interrupt in progress 1=High priority interrupt in progress	Read Only

**POWER MANAGEMENT AND STATUS BIT SUMMARY** Table 7-3

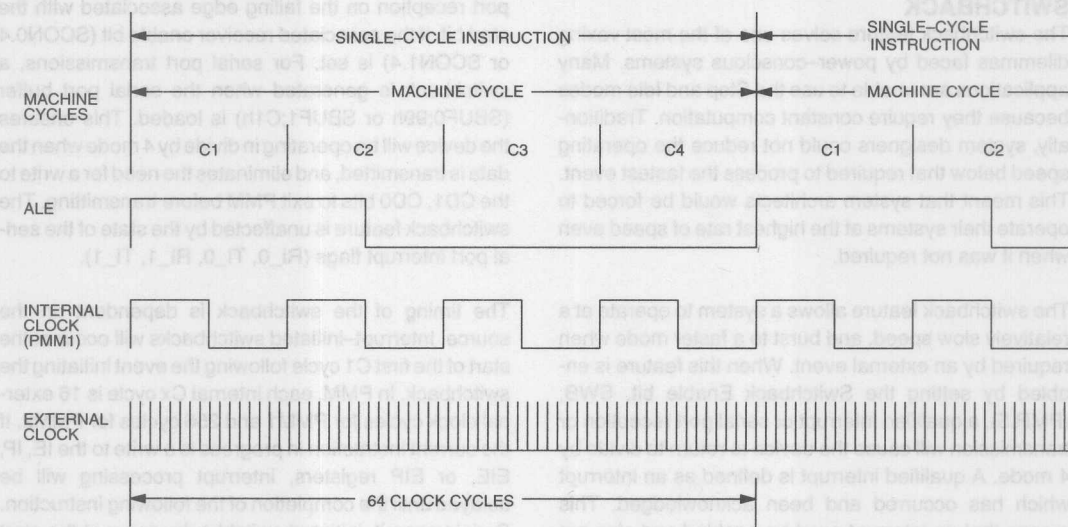
BIT NAME	LOCATION	FUNCTION	RESET STATE	READ/WRITE ACCESS
CD1, CD0	PMR.7-6	Clock Divider Control CD1 CD0 Osc Cycles per Machine Cycle 0 0 Reserved 0 1 4 (Reset Default) 1 0 64 (PMM1) 1 1 1024 (PMM2)	0,1	Write: 0,1 anytime; 1,0 & 1,1 only when previously in 0,1 state. Unrestricted read.
SWB	PMR.5	Switchback Enable 0=Interrupts and serial port activity will not affect clock divider control bits 1=Enabled Interrupts and serial port activity will cause a switchback	0	Unrestricted
PIP	STATUS.7	Power Fail Interrupt Status 0=No power fail interrupt in progress 1=Power fail interrupt in progress	0	Read Only
HIP	STATUS.6	High Priority Interrupt Status 0=No high priority interrupt in progress 1=High priority interrupt in progress	0	Read Only
LIP	STATUS.5	Low Priority Interrupt Status 0=No low priority interrupt in progress 1=Low priority interrupt in progress	0	Read Only
SPTA1	STATUS.3	Serial Port 1 Transmitter Activity Status 0=Serial port 1 transmitter inactive 1=Serial port 1 transmitter active	0	Read Only
SPRA1	STATUS.2	Serial Port 1 Receiver Activity Status 0=Serial port 1 receiver inactive 1=Serial port 1 receiver active	0	Read Only
SPTA0	STATUS.1	Serial Port 0 Transmitter Activity Status 0=Serial port 0 transmitter inactive 1=Serial port 0 transmitter active	0	Read Only
SPRA0	STATUS.0	Serial Port 0 Receiver Activity Status 0=Serial port 0 receiver inactive 1=Serial port 0 receiver active	0	Read Only

## POWER MANAGEMENT MODE TIMING

The two power management modes reduce power consumption by internally dividing the clock signal to the device, causing it to operate at a reduced speed. When PMM is invoked, the external crystal will continue to operate at full speed. The difference is that the device uses 16 (PMM1) or 256 (PMM2) external clocks to generate each internal clock cycle (C1, C2, C3 or C4) as opposed to 1 clock per internal clock cycle in divide by 4 mode. This translates to 64 or 1024 external clocks per machine cycle in PMM1 or PMM2, respectively. Rela-

tive timing relationships of all signals when the device is operating in PMM1 or PMM2 will remain the same as the 4 cycle timing. Note that all internal functions, on-board timers (including serial port baud rate generation), watchdog timer, and software timing loops will also run at the reduced speed. Most applications will not find it necessary to attend to this much detail, but the information is provided for calculating critical timings. Figure 7-2 demonstrates the internal timing relationships during PMM1.

## INTERNAL TIMING RELATIONSHIPS IN PMM1 Figure 7-2



PMM1 and PMM2 are entered and exited by setting the Clock Rate Divider bits (PMR.7-6). In addition, it is possible use the switchback feature to effect a return to the divide by 4 mode from either power management mode. This allows both hardware and software to cause an exit from PMM. Entry to or exit from either PMM must be by the divide by 4 mode. This means that to switch from divide by 64 to divide by 1024 and vice versa, one must first switch back to divide by 4 mode. Attempts to execute an illegal speed change will be ignored and the bits will remain unchanged. It is the responsibility of the software to test for serial port activity before attempting to change speed, as a modification of the clock divider bits during a serial port operation will corrupt the data.

## PMM AND PERIPHERAL FUNCTIONS

Timers 0, 1, and 2 will default on reset to a 12 clock per cycle operation to remain compatible with the original 8051 timing. The timers can be individually configured to run at machine cycle timing (divide by 4) by setting the relevant bits in the Clock Control Register (CKCON;8Eh). Because the timers derive their time base from the internal clock, timers 0, 1, and 2 operate at reduced clock rates during PMM. This will also affect the operation of the serial ports in PMM. In general, it is not possible to generate standard baud rates while in PMM, and the user is advised to avoid PMM or use the switchback feature if serial port operation is desired. Table 7-4 shows the effect of the clock divider value on timer operation.

**EFFECT OF CLOCK MODES ON TIMER OPERATION** Table 7-4

CD1	CD0	OSC. CYCLES PER MACHINE CYCLE	OSC. CYCLES PER TIMER 0/1/2 CLOCK		OSC. CYCLES PER TIMER 2 CLOCK, BAUD RATE GEN.		OSC. CYCLES PER SERIAL PORT CLOCK MODE 0		OSC. CYCLES PER SERIAL PORT CLOCK MODE 2	
			TxM=1	TxM=0	T2M=1	T2M=0	SM2=0	SM2=1	SMOD=0	SMOD=1
0	0	Reserved								
0	1	4	12	4	2	2	12	4	64	32
1	0	64 (PMM1)	192	64	32	32	192	64	1024	512
1	1	1024 (PMM2)	3072	1024	512	512	3072	1024	16,384	8192

**SWITCHBACK**

The switchback feature solves one of the most vexing dilemmas faced by power-conscious systems. Many applications are unable to use the Stop and Idle modes because they require constant computation. Traditionally, system designers could not reduce the operating speed below that required to process the fastest event. This meant that system architects would be forced to operate their systems at the highest rate of speed even when it was not required.

The switchback feature allows a system to operate at a relatively slow speed, and burst to a faster mode when required by an external event. When this feature is enabled by setting the Switchback Enable bit, SWB, (PMR.5), a qualified interrupt or serial port reception or transmission will cause the device to return to divide by 4 mode. A qualified interrupt is defined as an interrupt which has occurred and been acknowledged. This means that an interrupt must be enabled and also not blocked by a higher priority interrupt. After the event is complete, software can manually return the device to the appropriate PMM. The following sources can trigger a switchback:

- external interrupt 0/1/2/3/4/5,
- serial start bit detected, Serial Port 0/1,
- transmit buffer loaded, Serial Port 0/1,
- watchdog timer reset,
- power-on reset,
- external reset.

In the case of a serial port-initiated switchback, the switchback is not generated by the associated interrupt. This is because a device operating in PMM will not be able to correctly receive a byte of data to generate an interrupt. Instead, a switchback is generated by a serial

port reception on the falling edge associated with the start bit, if the associated receiver enable bit (SCON0.4 or SCON1.4) is set. For serial port transmissions, a switchback is generated when the serial port buffer (SBUF0;99h or SBUF1;C1h) is loaded. This ensures the device will be operating in divide by 4 mode when the data is transmitted, and eliminates the need for a write to the CD1, CD0 bits to exit PMM before transmitting. The switchback feature is unaffected by the state of the serial port interrupt flags (RI\_0, TI\_0, RI\_1, TI\_1).

The timing of the switchback is dependent on the source. Interrupt-initiated switchbacks will occur at the start of the first C1 cycle following the event initiating the switchback. In PMM, each internal Cx cycle is 16 external clock cycles for PMM1 and 256 cycles for PMM2. If the current instruction in progress is a write to the IE, IP, EIE, or EIP registers, interrupt processing will be delayed until the completion of the following instruction. Serial transmit-initiated switchbacks occur at the start of the instruction following the MOV that loads SBUF0 or SBUF1. Serial reception-initiated switchbacks occur during the Cx cycle in which the falling edge was detected.

There are a few points that must be considered when using a serial port reception to generate a switchback. Under normal circumstances, noise on the line or an aborted transmission would cause the serial port to time-out and the data to be ignored. This presents a problem if the switchback is used, however, because a switchback would occur but there is no indication to the system that one has occurred. If PMM and serial port switchback functions are used in a noisy environment, the user is advised to periodically check if the device has accidentally exited PMM.

A similar problem can occur if multiprocessor communication protocols are used in conjunction with PMM. The High-Speed Microcontroller family supports both the use of the SM2 flag (SCON0.5 or SCON1.5), and the slave address recognition registers (SADDR0;A9h, SADDR1;AAh, SADEN0;B9h, SADEN1;BAh) for multiprocessor communications. The problem is that an invalid address which should be ignored by a particular processor will still generate a switchback. As a result it is not recommended to use a multiprocessor communication scheme in conjunction with PMM. If the system power considerations will allow for an occasional erroneous switchback, a polling scheme can be used to place the device back into PMM.

### CLOCK SOURCE SELECTION

The High-Speed Microcontroller family supports three different clock sources for operation. As with most microcontrollers, the device can be clocked from an external crystal using the on-board crystal amplifier, or a clock source can be supplied by an external oscillator. In addition, some members of the High-Speed Microcontroller family incorporate an on-board ring oscillator to provide a quick resumption from Stop mode. The ring

oscillator is a low power digital oscillator internal to the microcontroller. When enabled, it provides an approximately 2–4 MHz clock source for device operation without external components. The ring oscillator is not as stable as an external crystal, and software should refrain from performing timing dependent operations, including serial port activity, while operating from the ring oscillator.

The ring oscillator provides many advantages to the designers of microcontroller-based systems. One is that it allows Dallas Semiconductor microcontrollers to perform a fast resume from Stop mode, eliminating the crystal warm-up delay when restarting the device. As an added feature, the DS87C520 and DS87C530 will also support extended operation from the ring oscillator, not only during the crystal warm-up period when resuming from Stop. All devices in the High-Speed Microcontroller family must begin operation following a power-on reset from an external clock source, either an external crystal or oscillator. Software can then disable the crystal and run from the lower power ring oscillator. The control and status bits which support the new and/or enhanced features are shown in Table 7–5.

XTUP	STATUS.4	Crystal Oscillator Warm Up Status. This bit is not present on the DS87C520.	None
		0-Crystal warm up still in progress, crystal not available	
		1-Crystal warm up complete	



XT/RG	EXIF.3	Crystal/Ring Clock Source Select. This bit is not present on the 80C320. 1=Select crystal or external clock as clock source, 0=Select ring oscillator as clock source	1	0 anytime; 1 when XTUP=1 & XTOFF=0
RGMD	EXIF.2	Ring Oscillator Mode Status. 1=Ring oscillator is current clock source, 0=Crystal or external clock is current clock source.	0	None
RGSL	EXIF.1	Ring Oscillator Select, Stop Mode. 1=Ring oscillator will be the clock source when resuming from Stop mode, 0=Crystal or external clock will be the clock source when resuming from Stop mode Note: Upon completion of crystal warm up period, DS80C320 devices will switch to crystal. DS87C520 and DS87C530 devices will switch to clock source designated by XT/RG bit		Unrestricted
XTOFF	PMR.3	Crystal Oscillator Disable. Disables crystal operation after ring mode has been selected. This bit is not present on the 80C320. 1=Crystal amplifier is disabled. 0=Crystal amplifier is enabled. Check XTUP for status.	0	0 anytime; 1 when XT/RG =0
XTUP	STATUS.4	Crystal Oscillator Warm Up Status. This bit is not present on the 80C320. 1=Oscillator warm up complete. 0=Oscillator warm up still in progress, crystal not available.	1	None

## RING OSCILLATOR RESUME FROM STOP

To achieve the minimum power consumption during periods of processor inactivity, software can place the device into Stop mode. Such systems will typically resume operation using an external interrupt, perform some activity, and then return to Stop mode. Traditional designs which rely upon an external crystal as the clock source must incur the startup delay of the crystal when resuming from Stop mode. This is a waste of time and power as no work can be performed until the crystal has stabilized.

Although the ring oscillator provides an approximately 2–4 MHz clock source for device operation, it is not as stable as an external crystal. As a result, high accuracy timing operations should be avoided while running from the ring oscillator. This includes using the timers for pulse measurement, and the use of the serial ports in asynchronous modes (1, 2, 3). Serial ports operating in mode 0 are unaffected by the stability of the clock source as a separate synchronizing clock is generated.

If the Ring Oscillator Select bit, RGSL (EXIF.1) is set, the device will resume operation immediately using the internal ring oscillator as the clock source. The device will continue to run from the ring oscillator until the crystal warm-up period of 65,536 clock cycles (measured from the external source) has completed. At this time the device will switch to the clock source active before it entered Stop mode and continue operation. This allows software execution to begin immediately upon resuming from Stop mode. The current clock source is indicated by the Ring Oscillator Mode bit, RGMD (EXIF.2).

In Stop mode, enabled interrupts become true edge triggered interrupts, compared with the sampled edge detection used during normal operation. This means that external interrupts are more sensitive to noise in Stop mode than during normal operation. Applications should be carefully designed to ensure that noise will not cause an erroneous exit from Stop mode.

## SWITCHING BETWEEN CLOCK SOURCES

DS87C520 and DS87C530 incorporate the ability to run the device from the ring oscillator after the crystal warm-up period has elapsed. Immediately following a reset (including initial power-up), all devices must operate from an external crystal or oscillator. At this point, software may switch to the ring oscillator by clearing the XT/ $\overline{\text{RG}}$  bit (EXIF.3). If there is no expectation that the crystal oscillator will be needed soon, the crystal oscillator can be disabled by setting the Crystal Oscillator Disable Bit, XT $\overline{\text{OFF}}$  (PMR.3). Note that switching to the ring oscillator does not automatically disable the crystal amplifier, and thus it is possible to be operating the device from the ring oscillator and have the external crystal amplifier operating at the same time. In some cases this may be desired to take advantage of the low-frequency, low-power feature of the ring oscillator but still have the capability of quickly switching back to the external crystal to perform timing or serial port operations.

Switching from the ring oscillator to the crystal oscillator is more involved due to the startup delays inherent in the external crystal. To prevent an accidental disabling of the device, the XTUP bit must be set by internal hardware (indicating an enabled, stable crystal) before setting the XT/ $\overline{\text{RG}}$  bit. The procedure to switch to the crystal oscillator when running from the ring oscillator is as follows:

1. Clear the Crystal Oscillator Disable Bit, XT $\overline{\text{OFF}}$  (PMR.3) to restart the crystal oscillator and start the crystal warm-up period.
2. Wait for the Crystal Oscillator Warm Up Status bit, XTUP (STATUS.4) to be set, indicating that the external crystal warm up period is complete. This will take 65,536 external clock cycles.
3. Set the Crystal Oscillator/Ring Oscillator Select Bit, XT/ $\overline{\text{RG}}$  (EXIF.3) to select the crystal as the clock source.

## SECTION 8: RESET CONDITIONS

The High-Speed Microcontroller provides several ways to place the CPU in a reset state. It also offers the means for software to determine the cause of a reset. The reset state of most processor bits is not dependent on the type of reset, but selected bits do depend on the reset source. The reset sources and the reset state are described below.

### RESET SOURCES

The High-Speed Microcontroller has three ways of entering a reset state. Each reset source is described below. They are:

- Power-on/Power Fail Reset
- Watchdog Timer Reset
- External Reset

#### Power-on/Fail Reset

Members of the High-Speed Microcontroller family incorporate an internal voltage reference which holds the CPU in the power-on reset state while  $V_{CC}$  is out of tolerance. Once  $V_{CC}$  has risen above the threshold, the microcontroller will restart the oscillation of the external crystal and count 65536 clock cycles. The processor will then begin software execution at location 0000h.

The voltage at which the reset state is entered depends on the specific device. If the device does not contain a precision voltage reference, the power-on reset threshold may be anywhere between 0.8V and  $V_{CCMIN}$ . If the device incorporates a precision voltage reference, the threshold will be as specified by the  $V_{RST}$  parameter in the data sheet. This helps the system maintain reliable operation by only permitting processor operation when voltage is in a known good state.

The processor will exit the reset condition automatically once the above conditions are met. This happens automatically, needing no external components or action. Execution begins at the standard reset vector address of 0000h. Software can determine that a Power-on Reset has occurred using the Power-on Reset flag (POR). It is located at WDCON.6. Since all resets cause a vector to location 0000h, the POR flag allows software to acknowledge that power failure was the reason for a reset.

Software should clear the POR bit after reading it. When a reset occurs, software will be able to determine if a power cycle was the cause. In this way, processing may

take a different course for each of the three resets if applicable. When power fails (drops below  $V_{RST}$ ), the power monitor will invoke the reset state again. This reset condition will remain while power is below the threshold. When power returns above the reset threshold, a full power-on reset will be performed. Thus a brownout that causes  $V_{CC}$  to drop below  $V_{RST}$  appears the same as a power up.

#### Watchdog Timer Reset

The Watchdog Timer is a free running timer with a programmable interval. Software can clear the timer at any time, causing the interval to begin again. The Watchdog supervises CPU operation by requiring software to clear it before the time-out expires. If the timer is enabled and software fails to clear it before this interval expires, the CPU is placed into a reset state. The reset state will be maintained for two machine cycles. Once the reset is removed, the software will resume execution at 0000h.

The Watchdog Timer is fully described in Section 11. Software can determine that a Watchdog time-out was the reason for the reset by using the Watchdog Timer Reset flag (WTRF). WTRF is located at WDCON.2. Hardware will set this bit to a logic 1 when the Watchdog times out without being cleared by software if EWTF=1. If a Watchdog Timer reset occurs, software should clear this flag manually. This allows software to detect the event if it occurs again.

#### External Reset

If the RST input is taken to a logic 1, the CPU will be forced into a reset state. This will not occur instantaneously, as the condition must be detected and then clocked into the microcontroller. It requires a minimum of two machine cycles to detect and invoke the reset state. Thus the reset is a synchronous operation and the crystal must be running to cause an external reset.

Once the reset state is invoked, it will be maintained as long as RST=1. When the RST is removed, the CPU will exit the reset state within two machine cycles and begin execution at address 0000h. All registers will default to their power-on reset state. There is no flag to indicate that an external reset was applied. However, since the other two sources have associated flags, the RST pin is the default source when neither POR or WTRF is set.

If a RST is applied while the processor is in the Stop mode, the scenario changes slightly. As mentioned above, the reset is synchronous and requires a clock to

be running. Since the Stop mode stops all clocks, the RST will first cause the oscillator to begin running and force the program counter to 0000h. Rather than a two machine cycle delay as described above, the processor will apply the full power-on delay (65536 clocks) to allow the oscillator to stabilize.

### RESET STATE

Regardless of the source of the reset, the state of the microcontroller is the same while in reset. When in reset, the oscillator is running, but no program execution is allowed. When the reset source is external, the user must remove the reset stimulus. When power is applied to the device, the power-on delay removes the stimulus automatically.

Resets do not affect the Scratchpad RAM. Thus any data stored in RAM will be preserved. The contents of internal MOVX data memory will also remain unaffected by a reset. Note that if the power supply dips below approximately 2V, the RAM contents may be lost. The minimum voltage required for RAM data retention is not specified. Since it is impossible to determine if the power was lower than 2V prior to the power-on reset, RAM must be assumed lost when POR is set.

The reset state of SFR bits are described in Section 4. Bits which are marked SPECIAL have conditions which can affect their reset state. Consult the individual bit descriptions for more information. Note that the stack pointer will also be reset. Thus the stack is effectively lost during a reset even though the RAM contents are not altered. Interrupts and Timers are disabled. The state of the Watchdog Timer is dependent on the specific device in use. Note that the Watchdog time out defaults to its shortest interval on any reset. I/O Ports are taken to a weak high state (FFh). This leaves each port pin configured with the data latch set to a 1. Ports do not go to the 1 state instantly when a reset is applied, but will be taken high within two machine cycles of asserting a reset. When the reset stimulus is removed, program execution begins at address 0000h.

### NO-BATTERY RESET

The battery backup feature of the DS87C530 introduces a new type of reset condition. Most SFR bits are automatically reset to their default state upon a power-on reset. The external backup battery feature makes

some bits non-volatile, however, and these battery-backed bits will not change state when a power-on reset is applied. Upon the loss or initial connection of battery power these bits will default to the state shown in Table 8-1. Any bits not listed below are either unchanged or set to their default state by a power-on reset.

**NO-BATTERY RESET DEFAULT** Table 8-1

BIT NAME	LOCATION	NO-BATTERY RESET STATE
E4K	TRIM.7	0
X12/6	TRIM.6	1
TRM2	TRIM.5	1
TRM2	TRIM.4	0
TRM1	TRIM.3	0
TRM1	TRIM.2	1
TRM0	TRIM.1	0
TRM0	TRIM.0	1
RTASS.7-0	RTASS.7-0	Indeterminate
RTAS.7-0	RTAS.7-0	Indeterminate
RTAM.7-0	RTAM.7-0	Indeterminate
RTAH.7-0	RTAH.7-0	Indeterminate
SSCE	RTCC.7	Indeterminate
SCE	RTCC.6	Indeterminate
MCE	RTCC.5	Indeterminate
HCE	RTCC.4	Indeterminate
RTCIF	RTCC.1	Indeterminate
RTCE	RTCC.0	Indeterminate
RTCSS.7-0	RTCSS.7-0	Indeterminate
RTCS.7-0	RTCS.7-0	Indeterminate
RTCM.7-0	RTCM.7-0	Indeterminate
RTCH.7-0	RTCH.7-0	Indeterminate
RTCD0.7-0	RTCD1.7-0	Indeterminate
RTCD1.7-0	RTCD1.7-0	Indeterminate

## IN-SYSTEM DISABLE MODE

The High Speed Microcontroller Family supports in-circuit debugging of designs. The In-System Disable (ISD) feature allows the device to be tristated for in-circuit emulation or board testing. During ISD mode, the device pins will take on the following states:

DEVICE PIN	STATE DURING ISD
Port 0, 1, 2, 3 RST, $\overline{EA}$	True Tristate
ALE, $\overline{PSEN}$	Weak Pull-up ( $\sim 10K\Omega$ )
XTAL1, XTAL2	Oscillator remains active

The following procedure is used to enter ISD mode:

1. Assert reset by pulling RST high,
2. Pull ALE low and pull  $\overline{PSEN}$  high,

3. Verify that P2.7, P2.6, P2.5 are not being driven low,
4. Release RST,
5. Hold ALE low and  $\overline{PSEN}$  high for at least 2 machine cycles,
6. Device is now in ISD mode. Release ALE and  $\overline{PSEN}$  if desired.

Note that pins P2.7, P2.6, P2.5 should not be driven low when RST is released. This will place the device into a reserved test mode. Because these pins have a weak pull-up during reset, they can be left floating. The test mode is only sampled on the falling edge of RST, and once RST is released their state will not effect device operation. In a similar manner, the  $\overline{PSEN}$  and RST pins can be released once ISD mode is invoked, and their state will not effect device operation. The RST pin will also be in a tristate mode, but asserting it in ISD mode will return the device to normal operation.

			TRM5
			TRM5
1. Assert reset by pulling RST high,		TRM1	TRM1
2. Pull ALE low and pull $\overline{\text{PSEN}}$ high,		TRM1	TRM1
	0	TRM1	TRM1
	1	TRM1	TRM1
	0	TRM1	TRM1
	1	TRM1	TRM1
		RTAS7-0	RTAS7-0
		RTAS7-0	RTAS7-0
		RTAM7-0	RTAM7-0
		RTAH7-0	RTAH7-0
		RTCC7	RTCC7
		RTCC8	RTCC8
		RTCC9	RTCC9
		RTCC4	RTCC4
		RTCC1	RTCC1
		RTCC0	RTCC0
		RTCS7-0	RTCS7-0
		RTCS7-0	RTCS7-0
		RTCM7-0	RTCM7-0
		RTCH7-0	RTCH7-0
		RTCD07-0	RTCD07-0
		RTCD17-0	RTCD17-0



## SECTION 9: INTERRUPTS

The High-Speed Microcontroller family utilizes a three-priority interrupt system. The number of interrupts varies according to the specific device. Each source has an independent priority bit, flag, interrupt vector, and enable. In addition, interrupts can be globally enabled (or disabled). The system is compatible with the original 8051 family. All of the original interrupts are available.

Several new sources have been added with new associated control and status bits, and new interrupt vectors. Note that the interrupt vector table can extend from 0000h to 006Bh, so existing code may require a re-location of the start address to avoid a conflict with the upper end of the vector table. A summary of all interrupts appears in Table 9-1 below.

**INTERRUPT SUMMARY** Table 9-1

INTERRUPT	INTERRUPT VECTOR	NATURAL PRIORITY	FLAG	ENABLE	PRIORITY CONTROL
Power-fail Indicator	33h	0	PFI (WDCON.4)	EPFI (WDCON.5)	N/A
External Interrupt 0	03h	1	IE0 (TCON.1)**	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0Bh	2	TF0 (TCON.5)*	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1	13h	3	IE1 (TCON.3)**	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	1Bh	4	TF1 (TCON.7)*	ET1 (IE.3)	PT1 (IP.3)
Serial Port 0	23h	5	RI_0 (SCON0.0), TI_0 (SCON0.1)	ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow	2Bh	6	TF2 (T2CON.7)	ET2 (IE.5)	PT2 (IP.5)
Serial Port 1	3Bh	7	RI_1 (SCON1.0), TI_1 (SCON1.1)	ES1 (IE.6)	PS1 (IP.6)
External Interrupt 2	43h	8	IE2 (EXIF.4)	EX2 (EIE.0)	PX2 (EIP.0)
External Interrupt 3	4Bh	9	IE3 (EXIF.5)	EX3 (EIE.1)	PX3 (EIP.1)
External Interrupt 4	53h	10	IE4 (EXIF.6)	EX4 (EIE.2)	PX4 (EIP.2)
External Interrupt 5	5Bh	11	IE5 (EXIF.7)	EX5 (EIE.3)	PX5 (EIP.3)
Watchdog Interrupt	63h	12	WDIF (WDCON.3)	EWDI (EIE.4)	PWDI (EIP.4)
Real-Time Clock	6Bh	13	RTCIF (RTCC.1)	ERTCI (EIE.5)	PRTCI (EIP.5)

Unless marked these flags must be cleared manually by software.

\* Cleared automatically by hardware when the service routine is vectored to.

\*\* If edge triggered, cleared automatically by hardware when the service routine is vectored to. If level triggered, flag follows the state of the pin.

## INTERRUPT OVERVIEW

An interrupt allows the software to react to unscheduled or asynchronous events. When an interrupt occurs, the CPU is expected to "service" the interrupt. This service takes the form of an Interrupt Service Routine (ISR). The ISR resides at a predetermined address as shown in Table 9-1. When the interrupt occurs, the CPU will vector to the appropriate location. It will run the code found at this location, staying in an interrupt service state until done with the ISR. Once an ISR has begun, it

can be interrupted only by a higher priority interrupt. The ISR is terminated by a return from interrupt instruction (RETI). When an RETI is performed, the processor will return to the instruction that would have been next when the interrupt occurred.

Each interrupt source has an associated vector. This is the address to which the CPU will jump when the interrupt occurs. When the interrupt condition occurs, the processor will also indicate this by setting a flag bit. This

bit is set regardless of whether the interrupt is enabled or not. That is, the flag responds to the condition, not the interrupt. Most flags must be cleared manually by software. However, IE0 and IE1 are cleared automatically by hardware when the service routine is vectored to if the interrupt was edge triggered. In level triggered mode, the flag follows the state of the pin. Flags TF0 and TF1 are always cleared automatically when the service routine is vectored to. Refer to the individual bit descriptions for more details. In order for the processor to acknowledge the interrupt and vector to the ISR, the interrupt must be enabled. Each source has an independent enable as shown in Table 9-1.

Prior to using any source, interrupts must be globally enabled. This is done using the EA bit at location IE.7. Setting this bit to a logic 1 allows individual interrupts to be enabled. Setting it to a logic 0 disables all interrupts regardless of the individual interrupt enables. The only exception is the Power-fail Interrupt. This is subject to its individual enable only. The EA bit has no effect on the Power-fail Interrupt.

## INTERRUPT SOURCES

Various combinations of interrupt sources are available on different members of the High-Speed Microcontroller family. These are broken into several categories: External, Timer-based, Serial Communication, real-time clock and Power Monitor. Each type is described below. Interrupt sources are sampled once per machine cycle. If the source goes active after the sample, it will not be registered until the next cycle.

### External Interrupts

The High-Speed Microcontroller has 6 external interrupt sources. These include the standard 2 interrupts of the 8051 architecture and 4 new sources. The original interrupts are  $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$ . These are active low, but can be programmed to be edge- or level-sensitive. The detection mode is controlled by bits IT0 and IT1, respectively. When ITx=0, the interrupt is triggered by a logic 0 on the appropriate interrupt pin. The interrupt condition remains in force as long as the pin is low. When ITx=1, the interrupt is pseudo edge-triggered. This means that if on successive samples, the pin is high then low, the interrupt is activated.

Since the external interrupts are sampled, the pin driver of an edge-triggered interrupt should hold the both the high, then the low condition for at least one machine cycles (each) to insure detection.

It is important to note that level-sensitive interrupts are not latched. If the interrupt is level-sensitive, the condition must be present until the processor can respond to the interrupt. This is most important if other interrupts are being used with a higher or equal priority. If the device is currently processing another interrupt, the condition must be present until the present interrupt is complete. This is because the level-sensitive interrupt will not be sampled until the RETI instruction is executed.

The remaining 4 external interrupts are similar in nature, with two differences. First, INT2 and INT4 are active high instead of active low. Second, all of the four new interrupts are edge-detect only. They do not have level-detect modes. All associated bits and flags operate the same and have the same polarity as the original two. A logic 1 indicates the presence of a condition, not the logic state of the pin.

If the Power Management Modes are utilized, the designer must remember that edge triggered interrupts must be high and low for one machine cycle before being recognized. This means that in PMM1 it will require 128 external clock cycles to recognize a level sensitive interrupt. Similarly, in PMM2 it will require 2048 external clock cycles to recognize a level sensitive interrupt. As a result, the interrupt latency for these interrupts will be slightly longer in PMM1 or PMM2.

### Timer Interrupts

The High-Speed Microcontroller incorporates three 16-bit programmable timers, each of which can generate an interrupt. In addition, some members of the family incorporate a programmable watchdog timer. The three programmable timers operate in the same manner as the 80C52. Each timer has an independent interrupt enable, flag, vector, and priority. The Watchdog Timer also has its own interrupt enable, flag, and priority.

Timers 0, 1, and 2 will set their respective flags when the timer overflows from a full condition, depending on its mode. This flag will be set regardless of the interrupt enable state. If the interrupt is enabled, this event will also cause a jump to the corresponding interrupt vector. For timers 0 and 1, the flags are cleared when the processor jumps to the interrupt vector. Thus these flags are not available for use by the interrupt service routine (ISR), but are available outside of the ISR and in applications that don't acknowledge the interrupt (i.e., jump to the vector). If the interrupt is not acknowledged, then software must manually clear the flag bit. In timer 2,

jumping to the interrupt vector does not clear the flag, so software must always clear it manually. Timer 0 and 1 flag bits reside in the TCON register. Timer 2 flag bit resides in the T2CON register. The interrupt enables and priorities for timers 0, 1, and 2 reside in the IE and IP registers respectively.

The Watchdog Interrupt usually has a different connotation than the Timer interrupts. Unless the Watchdog is being used as a very long timer, the interrupt means the software has failed to reset the counter and may be lost. The ISR can attempt to determine the system state. If the Watchdog is not cleared, the CPU will be reset in 512 clocks if EWT=1. Like other sources, the Watchdog Timer has a flag bit, an enable, and a priority. It also has its own vector. These are summarized in table 9-1.

### Serial Communication Interrupts

Each UART is capable of generating an interrupt. The UART has its own interrupt enable, vector, and priority. The UART differs from other sources as it has two flags. These are used by the ISR to determine whether the interrupt comes from a received word or a transmitted one. Unlike the timers, the UART flags are not altered when the interrupt is serviced. Software must change them manually.

When a UART finishes the transmission of a word, an interrupt will be generated (if enabled). Likewise, the UART will generate an interrupt when a word is completely received. The CPU will not be notified until the word is completely received or transmitted.

### Real-Time Clock

The DS87C530 real-time clock (RTC) has the ability to assert an RTC interrupt if enabled. The alarm can be programmed for a specific time once per day, or can be a recurring alarm once per hour, minute, second, or sub-second. This interrupt has the lowest priority of all interrupts, but can be used to bring the device out of Stop mode if desired. More information on this interrupt can be found in Section 14.

### Power-fail Interrupt

Some devices have the ability to generate an interrupt when  $V_{CC}$  drops below a predetermined level. These devices compare  $V_{CC}$  against an internal reference. If  $V_{CC}$  drops below the level  $V_{PFW}$ , an interrupt will result (if enabled). Note that the Power-fail Interrupt has the

highest priority. The priority level cannot be altered by the user, but the interrupt can be disabled if not needed. The level of  $V_{PFW}$  is provided in the data sheet specifications associated with each product. Note that the EPFI bit enables the Power-fail Interrupt. This bit is not subject to the global interrupt enable (EA). The Power-fail interrupt is a level-sensitive interrupt and will remain set as long as  $V_{CC}$  remains below  $V_{PFW}$ .

### Simulated Interrupts

Software can simulate any interrupt source by setting the corresponding flag bit. This forces an interrupt condition which will be acknowledged if enabled and is otherwise indistinguishable from the real thing. Thus an interrupt flag bit should never be set to a logic 1 by software inadvertently. Once an interrupt has been acknowledged, software cannot prevent or end the interrupt by clearing its flag. However, if for some reason the interrupt acknowledge is delayed, software may clear the flag and thereby prevent the interrupt from occurring. One exception is the real-time clock interrupt flag, RTCIF, which cannot be set in software.

### INTERRUPT PRIORITIES

The High-Speed Microcontroller has three interrupt priority levels. These are highest, high, and low. The Power-fail Interrupt is the only source that has highest priority and this level is fixed. The remaining sources are individually programmable to either high or low. Low priority is the default. A low priority interrupt can be interrupted by a high (or highest) priority interrupt. A high priority interrupt can only be interrupted by the Power-fail interrupt.

When an interrupt occurs and is serviced, its priority determines if its ISR can be interrupted. No interrupt source of equal or lesser priority can interrupt another source. That is, an incoming interrupt must be of a higher priority than the one currently being serviced to have priority.

If two interrupt sources of equal priority levels are requested simultaneously, the natural priority is used to arbitrate. The natural priority is given in Table 9-1. Note that natural priority is only used to resolve simultaneous requests. Once an interrupt of a given priority is invoked, only a source that is programmed with a higher priority can intercede.

multiple machine cycles that begin with the setting of the associated flag. For edge triggered external interrupts and internal interrupt sources, the interrupt flags are set automatically by hardware. For level sensitive external interrupts, the flags are actually under control of the external signal, and the flag will rise and fall with the pin level. Each interrupt flag is sampled once per machine cycle. Later in the same machine cycle, the samples are polled by hardware. If the sample indicates a pending interrupt and the interrupt is enabled, then on the next machine cycle it will be acknowledged by the hardware forcing an LCALL to the appropriate vector address. This LCALL will occur unless blocked by one of the following conditions.

1. An interrupt of equal or greater priority has already been invoked and the RETI instruction has not been issued to terminate it.
2. The current machine cycle is not the final cycle in the execution of the current instruction.
3. The instruction in progress is an RETI or an access to IP, IE, EIP, or EIE.

The individual interrupt sources and associated enable and priority bits are shown in Figure 9-1. While the final selection of the appropriate interrupt vector address is referred to as a polling process, this function is actually

## INTERRUPT LATENCY

Interrupt response will require a varying amount of time depending on the state of the microcontroller when the interrupt occurs. If the microcontroller is performing an ISR with equal or greater priority, the new interrupt will not be invoked. In other cases, the response time depends on the current instruction. The fastest possible response to an interrupt is 5 machine cycles. This includes one cycle for detecting the interrupt and four cycles to perform the LCALL that is inherent in the interrupt request. The maximum response time (if no other interrupt is in service) occurs if the microcontroller is performing an RETI instruction, and then executes a MUL or DIV as the next instruction. From the time an interrupt source is activated (not detected), the longest reaction time is 13 machine cycles. This includes 1 cycle to detect the interrupt, 3 cycles to finish the RETI, 5 to perform the MUL or DIV, then 4 for the LCALL to the ISR.

The maximum latency of 13 machine cycle is 52 clocks ( $13 \times 4$ ). Note that the maximum interrupt latency of an 8051 is 96 clocks (8 machine cycles @ 12 clocks per machine cycle). The maximum latency for the High-Speed Microcontroller at 25 MHz is about 2  $\mu$ s. The use of Power Management modes can further increase the interrupt latency.

When an interrupt occurs and is serviced, its priority is determined. If an ISR can be interrupted, its interrupt source of equal or lesser priority can interrupt another source. That is, an incoming interrupt must be of a higher priority than the one currently being serviced to have priority.

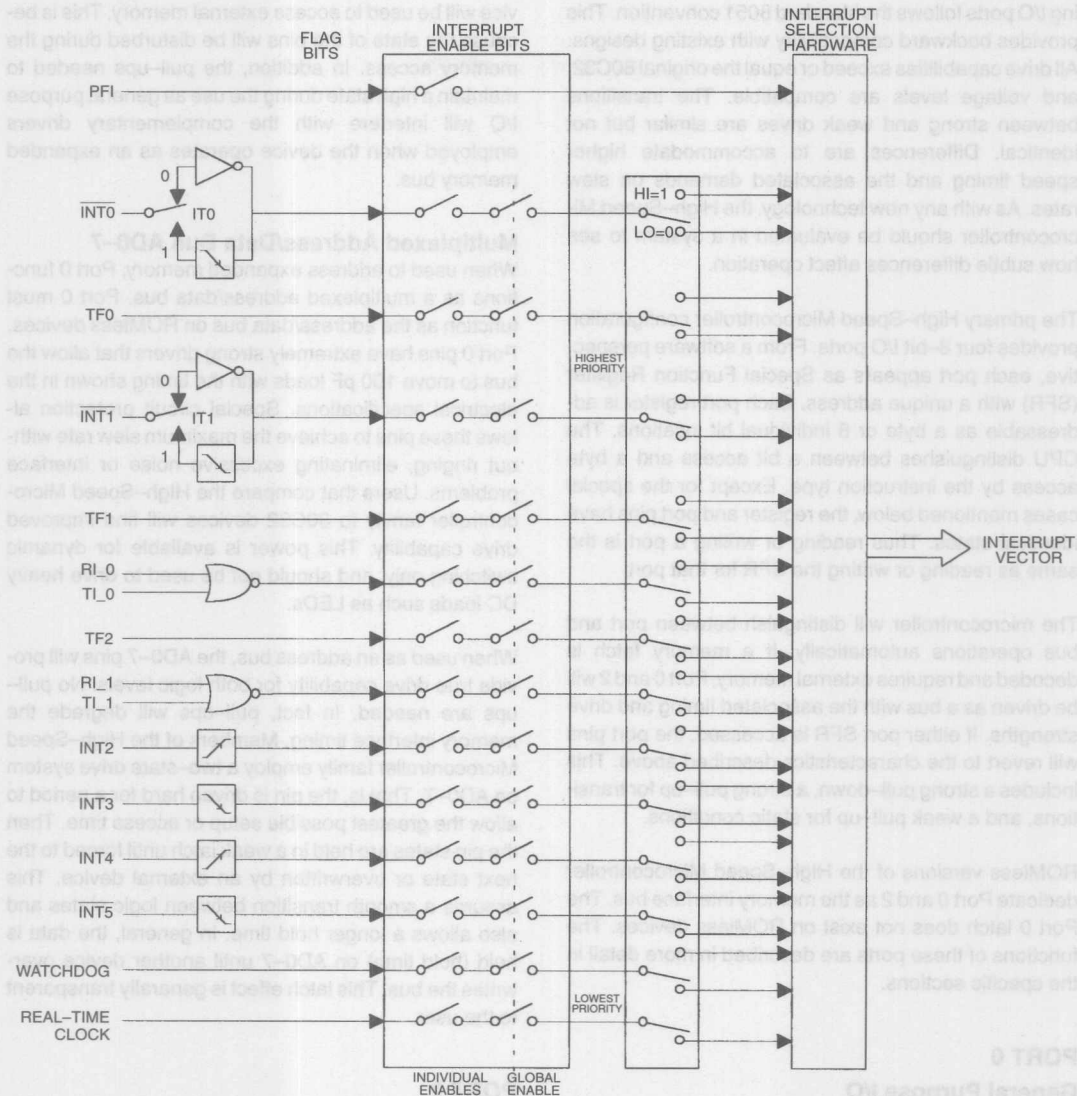
If two interrupt sources of equal priority levels are requested simultaneously, the natural priority is used to arbitrate. The natural priority is given in Table 9-1. Note that natural priority is only used to resolve simultaneous requests. Once an interrupt of a given priority is invoked, only a source that is programmed with a higher priority can interrupt.

**Real-Time Clock**  
The DS87C50 real-time clock (RTC) has the ability to assert an RTC interrupt if enabled. The alarm can be programmed for a specific time once per day, or can be a recurring alarm once per hour, minute, second, or sub-second. The interrupt has the lowest priority of all interrupts, but can be used to bring the device out of Sleep mode if desired. More information on this interrupt can be found in Section 14.

**Power-fail interrupt**  
Some devices have the ability to generate an interrupt when V<sub>CC</sub> drops below a predetermined level. These devices compare V<sub>CC</sub> against an internal reference. If V<sub>CC</sub> drops below the level V<sub>REF</sub>, an interrupt will occur (if enabled). Note that the Power-fail interrupt has the



# **INTERRUPT FUNCTIONAL DESCRIPTION Figure 9-1**





## SECTION 10: PARALLEL I/O

The High-Speed Microcontroller method of implementing I/O ports follows the standard 8051 convention. This provides backward compatibility with existing designs. All drive capabilities exceed or equal the original 80C32, and voltage levels are compatible. The transitions between strong and weak drives are similar but not identical. Differences are to accommodate higher speed timing and the associated demands on slew rates. As with any new technology, the High-Speed Microcontroller should be evaluated in a system to see how subtle differences affect operation.

The primary High-Speed Microcontroller configuration provides four 8-bit I/O ports. From a software perspective, each port appears as Special Function Register (SFR) with a unique address. Each port register is addressable as a byte or 8 individual bit locations. The CPU distinguishes between a bit access and a byte access by the instruction type. Except for the special cases mentioned below, the register and port pins have identical states. Thus reading or writing a port is the same as reading or writing the SFR for that port.

The microcontroller will distinguish between port and bus operations automatically. If a memory fetch is decoded and requires external memory, Port 0 and 2 will be driven as a bus with the associated timing and drive strengths. If either port SFR is accessed, the port pins will revert to the characteristics described above. This includes a strong pull-down, a strong pull-up for transitions, and a weak pull-up for static conditions.

ROMless versions of the High-Speed Microcontroller dedicate Port 0 and 2 as the memory interface bus. The Port 0 latch does not exist on ROMless devices. The functions of these ports are described in more detail in the specific sections.

### PORT 0

#### General Purpose I/O

Devices which have internal program memory have the ability to use Port 0 as a general purpose I/O. Data written to the port latch serves to set both level and direction of the data on the pin. ROMless devices do not contain a Port 0 latch, because at no time can it be manipulated as a port. When used as an I/O port, it functions as an open-drain output. More detail on the functions of these pins is provided under the description of output and input functions in this section.

Even if internal memory is present, the use of Port 0 as general purpose I/O pins is not recommended if the device will be used to access external memory. This is because the state of the pins will be disturbed during the memory access. In addition, the pull-ups needed to maintain a high state during the use as general purpose I/O will interfere with the complementary drivers employed when the device operates as an expanded memory bus.

#### Multiplexed Address/Data Bus AD0-7

When used to address expanded memory, Port 0 functions as a multiplexed address/data bus. Port 0 must function as the address/data bus on ROMless devices. Port 0 pins have extremely strong drivers that allow the bus to move 100 pF loads with the timing shown in the electrical specifications. Special circuit protection allows these pins to achieve the maximum slew rate without ringing, eliminating excessive noise or interface problems. Users that compare the High-Speed Microcontroller family to 80C32 devices will find improved drive capability. This power is available for dynamic switching only, and should not be used to drive heavy DC loads such as LEDs.

When used as an address bus, the AD0-7 pins will provide true drive capability for both logic levels. No pull-ups are needed. In fact, pull-ups will degrade the memory interface timing. Members of the High-Speed Microcontroller family employ a two-state drive system on AD0-7. That is, the pin is driven hard for a period to allow the greatest possible setup or access time. Then the pin states are held in a weak latch until forced to the next state or overwritten by an external device. This assures a smooth transition between logic states and also allows a longer hold time. In general, the data is held (hold time) on AD0-7 until another device overwrites the bus. This latch effect is generally transparent to the user.

### PORT 2

#### General Purpose I/O

Devices which have internal program memory have the ability to use Port 2 for a general purpose I/O. Data written to the port latch serves to set both level and direction of the data on the pin. When used as an I/O port, it has complementary outputs that will drive both high and low logic levels. More detail on the functions of these pins is provided under the description of output and input functions in this section.

Even if internal memory is present, the use of Port 2 as general purpose I/O pins is not recommended if the device will be used to access external memory. This is because the state of the pins will be disturbed during the memory access. It is still possible, however, to use the Port 2 latch to hold the upper address byte for Register Indirect Addressing instructions.

### Most Significant Address Byte, A8–15

When used to address expanded memory, Port 2 functions as the most significant byte of the address bus. Port 2 must function as the address bus on ROMless devices. When serving as a bus, Port 2 will be driven with strong drivers at all times except immediately after the rising edge of  $\overline{\text{PSEN}}$ . See Figure 5–3 and 5–4 for more details.

### PORTS 1 AND 3

Ports 1 and 3 are general purpose I/O ports with optional special functions associated with each pin. Enabling the special function automatically converts the I/O pin to that function. To insure proper operation, each alternate function pin should be programmed to a logic 1. For example, enabling the UART converts P3.0 and P3.1 to the serial I/O functions.

The drive characteristics of these pins do not change when the pin is configured for general I/O or as the special function associated with that pin. The exceptions are pins P3.6 and P3.7, which employ the current-limited transition drivers described later when used as  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  signals. The drive characteristics of Port 1 and Port 3 are the same as for Port 2 (non-bus mode). That is, the logic 0 is created by a strong pull-down. The logic 1 is created by a strong transition pull-up that changes to a weak pull-up.

Using one or more I/O pins of a port as special function pins will not effect the remaining port pins. An extreme example is as follows. P3.6 has the alternate function of  $\overline{\text{WR}}$  and P3.7 of  $\overline{\text{RD}}$ . These strobes are used for expanded data memory access. If a system used only the  $\overline{\text{RD}}$  signal, then P3.6 would still be available as an I/O port. This is not a practical suggestion, but it illustrates how the special functions are independent.

A more practical application is the optional use of an interrupt. If  $\overline{\text{INT0}}$  (P3.2) is enabled, then an externally imposed logic 0 will cause an interrupt. By then disabling the  $\overline{\text{INT0}}$ , P3.2 can be used as a general purpose I/O pin. This allows the  $\overline{\text{INT0}}$  to be used to "wake-up" the system, but does not eliminate another use of the pin.

### OUTPUT FUNCTIONS

Although 8051 I/O ports appear to be true I/O, their output characteristics are dependent on the individual port and pin conditions. When software writes a logic 0 to the port for output, the port is pulled to ground. When software writes a logic 1 to the port for output, Ports 1, 2, or 3 will drive weak pull-ups (after the strong transition from 0 to 1). Port 0 will go tri-state. Thus as long as the port is not heavily loaded, true logic values will be output. DC drive capability is provided in the electrical specifications. Note that the DC current available from an I/O port pin is a function of the permissible voltage drop.

Transition current is available to help move the port pin from a 0 to a 1. Since the logic 0 driver is strong, no additional drive current is needed in the 1 to 0 direction. The transition current is applied when the port latch is changed from a logic 0 to a logic 1. Simply writing a logic 1 where a 1 was already in place does not change the strength of the pull-up. This transition current is applied for a one half of a machine cycle. The absolute current is not guaranteed, but is approximately 2 mA at 5V.

When serving as an I/O port, the drive will vary as follows. For a logic 0, the port will invoke a strong pull-down. For a logic 1, the port will invoke a strong pull-up for two oscillator cycles to assist with the logic transition. Then, the port will revert to a weak pull-up. This weak pull-up will be maintained until the port transitions from a 1 to a 0. The weak pull-up can be overdriven by external circuits. This allows the output 1 state to serve as the input state as well.

Substantial DC current is available in both the high and low levels. However, the power dissipation limitations make it inadvisable to heavily load multiple pins. In general, sink and source currents should not exceed 10 mA total per port (8 bits) and 25 mA total per package.

special circuitry to limit the current consumed by the device when the expanded memory bus is used. These signals employ current-limited drivers which "step" the transition from a logic 0 to a logic 1 to reduce ringing and electromagnetic interference. When expanded memory operations are in progress, the following pins will exhibit the current-limiting feature:

Port 0  
Port 2  
PSEN (During program memory accesses)  
ALE  
RD (During data memory read cycles)  
WR (During data memory write cycles)

### INPUT FUNCTIONS

The input state of the I/O ports is the same as that of the output logic 1. That is, the pin is pulled weakly to a logic 1. This 1 state is easily overcome by external components. Thus, after software writes a 1 to the port pin, the port is configured for input. When the port is read by software, the state of the pin will be read. The only exception is the read-modify-write instructions described below. If the external circuit is driving a logic 1, then the pin will be a logic 1. If the external circuit is driving a 0, then it will overcome the internal pull-up. Thus the pin will be the same as the driven logic state. Note that the port latch is not altered by a read operation. Therefore, if a logic 0 is driven onto a port pin from an external source, then removed, the pin will revert to the weak pull-up as determined by the internal latch.

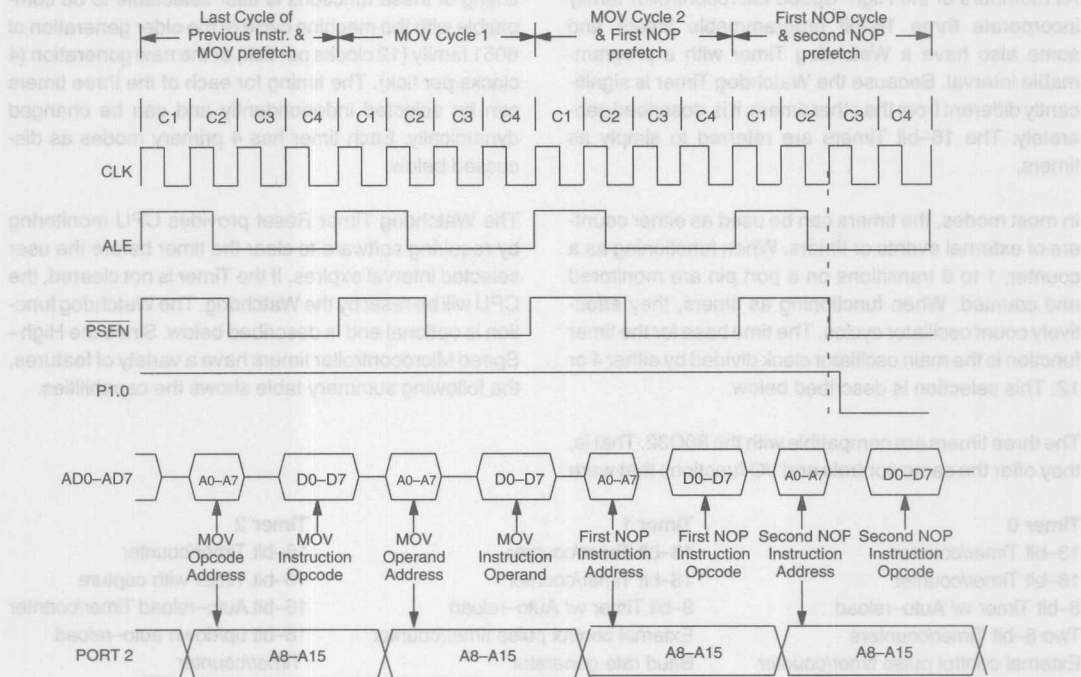
out regard to the output data latch. The only exception is the read-modify-write category of instructions. They are listed as follows.

INSTRUCTION	DESCRIPTION
ANL	Logical AND
ORL	Logical OR
XRL	Logical Exclusive OR (XOR)
JBC	Branch if bit set then clear bit
CPL	Complement bit
INC	Increment
DEC	Decrement
DJNZ	Decrement and branch if not zero
MOV PX.n, C	Move the carry bit to bit n of port X
CLR PX.n	Clear bit n of port X
SETB PX.n	Set bit n of port X

The read-modify-write instructions read the state of the latch, then write back the result to the latch. Thus the operation takes place using the value that was originally written to the SFR, without regard to the pin state. The last three instructions listed above are read-modify-write because they read the entire port latch, then write back the changed value. In this case, only one bit will be changed as specified by the instruction.

### I/O PORT TIMING

Figure 10-1 shows when port pins change in relationship to instruction timing. The example shown uses a MOV command to change P1.0 from a logic 1 to a logic 0. This diagram is presented to aid the designer in determining the timing relationship for very critical designs. Most designers will not need to consider this much detail.

**I/O PORT TIMING FOR MOV INSTRUCTION Figure 10-1****OPTIONAL FUNCTIONS**

Every port pin on the High-Speed Microcontroller has an optional special function. These functions are individually selectable. They can also be turned on and off dynamically to suit the application. The optional function for each port pin is described briefly below. More information about each optional function is available in the section dealing with that function.

P0.0	AD0.0 Multiplexed Address/data bus
P2.0	A8 MSB Address bus
P0.1	AD0.1 Multiplexed Address/data bus
P2.1	A9 MSB Address bus
P0.2	AD0.2 Multiplexed Address/data bus
P2.2	A10 MSB Address bus
P0.3	AD0.3 Multiplexed Address/data bus
P2.3	A11 MSB Address bus
P0.4	AD0.4 Multiplexed Address/data bus
P2.4	A12 MSB Address bus
P0.5	AD0.5 Multiplexed Address/data bus

P2.5	A13 MSB Address bus
P0.6	AD0.6 Multiplexed Address/data bus
P2.6	A14 MSB Address bus
P0.7	AD0.7 Multiplexed Address/data bus
P2.7	A15 MSB Address bus
P1.0	T2 Timer 2 output pulse
P3.0	RXD0 Serial Receive UART0
P1.1	T2EX Timer 2 capture/reload input
P3.1	TXD0 Serial Transmit UART0
P1.2	RXD1 Serial Receive UART1
P3.2	INT0 External Interrupt 0 active low
P1.3	TXD1 Serial Transmit UART1
P3.3	INT1 External Interrupt 1 active low
P1.4	INT2 External Interrupt 2 rising edge active
P3.4	T0 Timer 0 input
P1.5	INT3 External Int. 3 falling edge active
P3.5	T1 Timer 1 input
P1.6	INT4 External Interrupt 4 rising edge active
P3.6	WR Write strobe
P1.7	INT5 External Int. 5 falling edge active
P3.7	RD Read strobe



## SECTION 11: PROGRAMMABLE TIMERS

All members of the High-Speed Microcontroller family incorporate three 16-bit programmable timers and some also have a Watchdog Timer with a programmable interval. Because the Watchdog Timer is significantly different from the other timers, it is described separately. The 16-bit Timers are referred to simply as timers.

In most modes, the timers can be used as either counters of external events or timers. When functioning as a counter, 1 to 0 transitions on a port pin are monitored and counted. When functioning as timers, they effectively count oscillator cycles. The time base for the timer function is the main oscillator clock divided by either 4 or 12. This selection is described below.

The three timers are compatible with the 80C32. That is, they offer the same controls and I/O functions that were

available in the 80C32. As mentioned above, the actual timing of these functions is user selectable to be compatible with the machine cycle of the older generation of 8051 family (12 clocks per tick) or the new generation (4 clocks per tick). The timing for each of the three timers can be selected independently and can be changed dynamically. Each timer has 4 primary modes as discussed below.

The Watchdog Timer Reset provides CPU monitoring by requiring software to clear the timer before the user selected interval expires. If the Timer is not cleared, the CPU will be reset by the Watchdog. The Watchdog function is optional and is described below. Since the High-Speed Microcontroller timers have a variety of features, the following summary table shows the capabilities.

### Timer 0

13-bit Timer/counter  
16-bit Timer/counter  
8-bit Timer w/ Auto-reload  
Two 8-bit Timer/counters  
External control pulse timer/counter

### Timer 1

13-bit Timer/counter  
16-bit Timer/counter  
8-bit Timer w/ Auto-reload  
External control pulse timer/counter  
Baud rate generator

### Timer 2

16-bit Timer/counter  
16-bit Timer with capture  
16-bit Auto-reload Timer/counter  
16-bit up/down auto-reload  
Timer/counter  
Baud rate generator  
Timer output clock generator

## 16-BIT TIMERS

Timers 0 and 1 are nearly identical. Timer 2 has several additional features such as up/down counting, capture values and an optional output pin that make it unique. Timers 0 and 1 are described first. Timer 2 is described separately. As mentioned above, the time base for each timer can be varied and this is also discussed in more detail below.

Timer 0 and 1 both have four operating modes. They are 13-bit timer/counter, 16-bit timer/counter, 8-bit timer/counter with auto-reload, and two 8-bit timers. The latter mode is available to Timer 0 only. These modes are controlled by the TMOD register. Each timer can also

serve as a counter of external pulses (1 to 0 transition) on the corresponding Tn pin. This selection is controlled by the TMOD register. One other option is to gate the timer/counter using an external control signal. This allows the timer to measure the pulse width of external signals. Timers 0 and 1 are enabled using the TCON register which is also the location of their flags. The registers are described below. Following this is a detailed explanation of the four operating modes.

Each timer consists of a 16-bit register in two bytes. These are called TL0, TH0, TL1, and TH1. As shown, each timer is broken into low and high bytes. Software can read or write any of these locations at any time.



## TMOD REGISTER SUMMARY

## TIMER MODE CONTROL TMOD; 89h

TMOD.7 GATE – Timer 1 GATE control. When GATE=1, Timer 1 will clock only when  $\overline{\text{INT1}}$  and TR1 =1. When GATE=0, Timer 1 will clock only when TR1=1 irrespective of  $\overline{\text{INT1}}$ .

TMOD.6  $\text{C}/\overline{\text{T}}$  – Counter/Timer select. When  $\text{C}/\overline{\text{T}}$  is set to a 0, Timer 1 is incremented by internal clocks. When  $\text{C}/\overline{\text{T}}$  is set to a 1, Timer 1 counts based on the T1 (P3.5) pin.

TMOD.5 M1 – Timer 1 Mode select bit 1.

TMOD.4 M0 – Timer 1 Mode select bit 0.

M1	M0	Mode
0	0	Mode 0 : 8-bits with 5-bit prescale
0	1	Mode 1 : 16-bits
1	0	Mode 2 : 8-bits with auto-reload
1	1	Mode 3 : Timer 1 stopped

TMOD.3 GATE – Timer 0 GATE control. When GATE=1, Timer 0 will clock only when  $\overline{\text{INT0}}$  and TR0 =1. When GATE=0, Timer 0 will clock only when TR0=1 irrespective of  $\overline{\text{INT0}}$ .

TMOD.2  $\text{C}/\overline{\text{T}}$  – Counter/Timer select. When  $\text{C}/\overline{\text{T}}$  is set to a 0, Timer 0 is incremented by internal clocks. When  $\text{C}/\overline{\text{T}}$  is set to a 1, Timer 0 counts based on the T0 (P3.4) pin.

TMOD.1 M1 – Timer 0 Mode select bit 1.

TMOD.0 M0 – Timer 0 Mode select bit 0.

M1	M0	Mode
0	0	Mode 0 : 8-bits with 5-bit prescale
0	1	Mode 1 : 16-bits
1	0	Mode 2 : 8-bits with auto-reload
1	1	Mode 3 : Timer 0 is two 8-bit timers

## TIMER/COUNTER CONTROL

TCON; 88h

TCON.7	TF1 – Timer 1 overflow flag. Set to one when Timer 1 overflows from FFh, and cleared when the processor vectors to the interrupt service routine.
TCON.6	TR1 – Timer 1 run control. Turns on Timer 1 when this bit is set.
TCON.5	TF0 – Timer 0 overflow flag. Set to one when Timer 0 overflows from FFh, and cleared when the processor vectors to the interrupt service routine.
TCON.4	TR0 – Timer 0 run control. Turns on Timer 0 when this bit is set to a one.
TCON.3	IE1 – Interrupt 1 edge detect. Set by hardware when an edge/level is detected on $\overline{\text{INT1}}$ .
TCON.2	IT1 – Interrupt 1 type select. $\overline{\text{INT1}}$ detects a falling edge when this bit is set to a 1. $\overline{\text{INT1}}$ detects a low level when this bit is a 0.
TCON.1	IE0 – Interrupt 0 edge detect. Set by hardware when an edge/level is detected on $\overline{\text{INT0}}$ .
TCON.0	IT0 – Interrupt 0 type select. $\overline{\text{INT0}}$ detects a falling edge when this bit is set to a 1. $\overline{\text{INT0}}$ detects a low level when this bit is a 0.

### MODE 0

Mode 0 configures either Timer 0 or Timer 1 for operation as a 13-bit Timer/Counter. As shown in figure 11–1, bits M1=0 and M0=0 of the TMOD register select this operating mode.

When using Timer 0, TL0 uses only bits 0–4. These bits serve as the 5 LSBs of the 13-bit timer. TH0 provides the 8 MSBs of the 13-bit timer. Bit 4 of TL0 is used as a ripple out to TH0 bit 0, thereby completely bypassing bits 5 through 7 of TL0. Once the timer is started using the TR0 (TCON.4) timer enable, the timer will count as long as GATE (TMOD.3) is 0 or GATE is 1 and pin  $\overline{\text{INT0}}$  is 1. It will count oscillator cycles if  $\text{C}/\overline{\text{T}}$  (TMOD.2) is set to a logic 0 and 1 to 0 transitions on T0 (P3.4) if  $\text{C}/\overline{\text{T}}$  is set to a 1. When the 13-bit count reaches 1FFFh (all ones), the next count will cause it to roll over to 0000h. The TF0 (TCON.5) flag will be set and an interrupt will occur if enabled. The upper three bits of TL0 will be indeterminate.

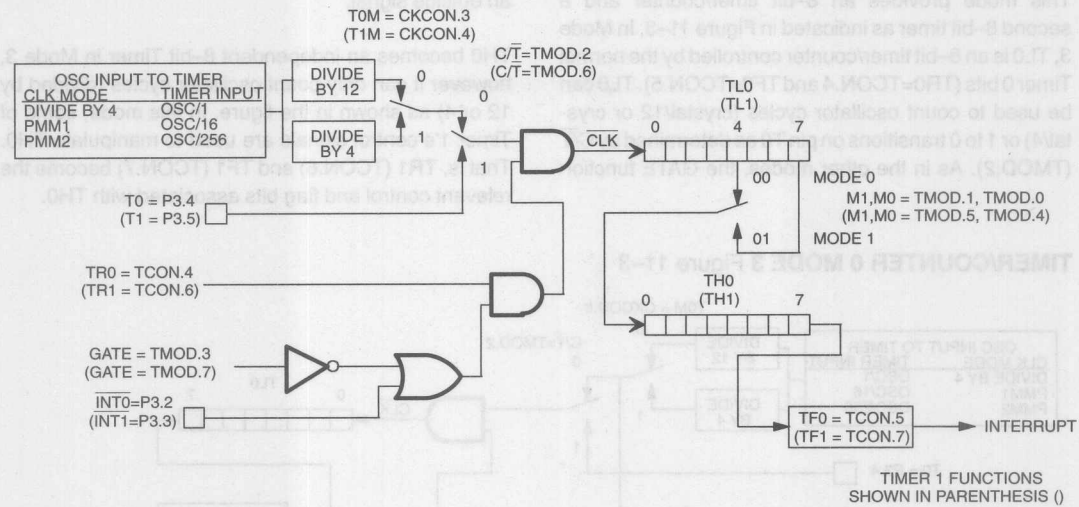
Note that when used as a timer, the time base may be either oscillator cycles/12 or oscillator cycles/4 as

selected by bits TnM (n=0 or 1) of the CKCON register. This feature is described in more detail below.

Mode 0 operates identically when Timer 1 is used. The same information applies to TL1 and TH1, which form the 13-bit register. TR1 (TCON.6),  $\overline{\text{INT1}}$  (P3.3), T1 (P3.5), and the relevant  $\text{C}/\overline{\text{T}}$  (TMOD.6) and GATE (TMOD.7) bits have the same functions.

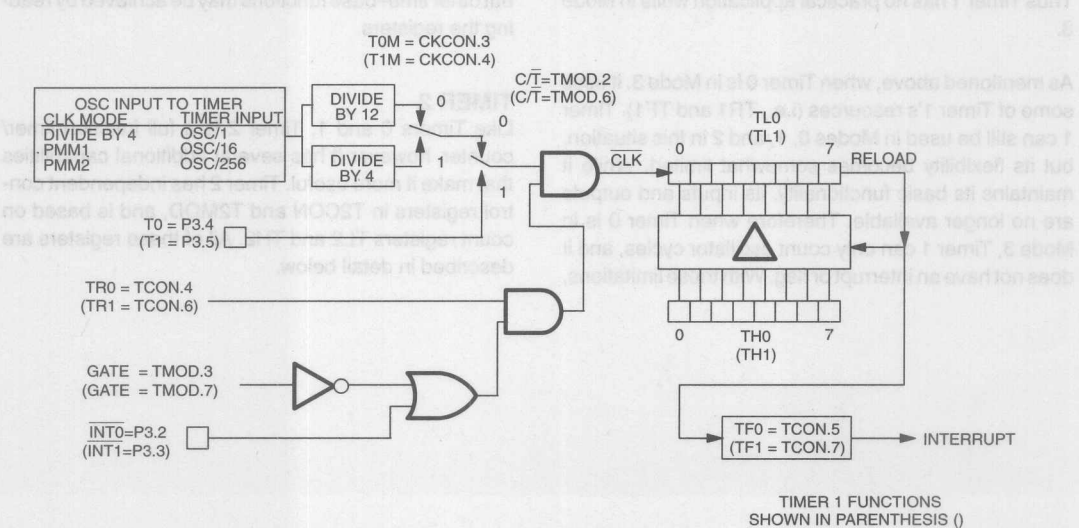
### MODE 1

Mode 1 configures the timer for 16-bit operation as either a timer or counter. Figure 11–1 shows that bits M1=0 and M0=1 of the TMOD register select this operating mode. For Timer n, all of the TLn and THn registers are used. For example, if Timer 1 is configured in mode 1, then TL1 holds the LSB and TH1 holds the MSB. Roll-over occurs when the timer reaches FFFFh. An interrupt will also occur if enabled and the relevant TFn flag is set. Timebase selection, counter/timer selection, and the gate function operate as described in mode 0.

**TIMER/COUNTER 0 AND 1 MODES 0 AND 1** Figure 11-1**MODE 2**

This mode configures the timer as an 8-bit timer/counter with automatic reload of the start value. This configuration is shown in Figure 11-2, and is selected when bits M1 and M0 of the TCON register are set to 1 and 0 respectively. When configured in Mode 2, the timer uses TLn to count and THn to store the reload value. Software must initialize both TLn and THn with the same starting value for the first count to be correct.

Once the TLn reaches FFh, it will be automatically loaded with the value in THn. The THn value remains unchanged unless modified by software. Mode 2 is commonly used to generate baud rates since it runs without continued software intervention. As in modes 0 and 1, mode 2 allows counting of either oscillator cycles (crystal/12 or crystal/4) or pulses on pin Tn ( $C/\bar{T}=1$ ) when counting is enabled by TRn and the proper setting of GATE and INTn pins.

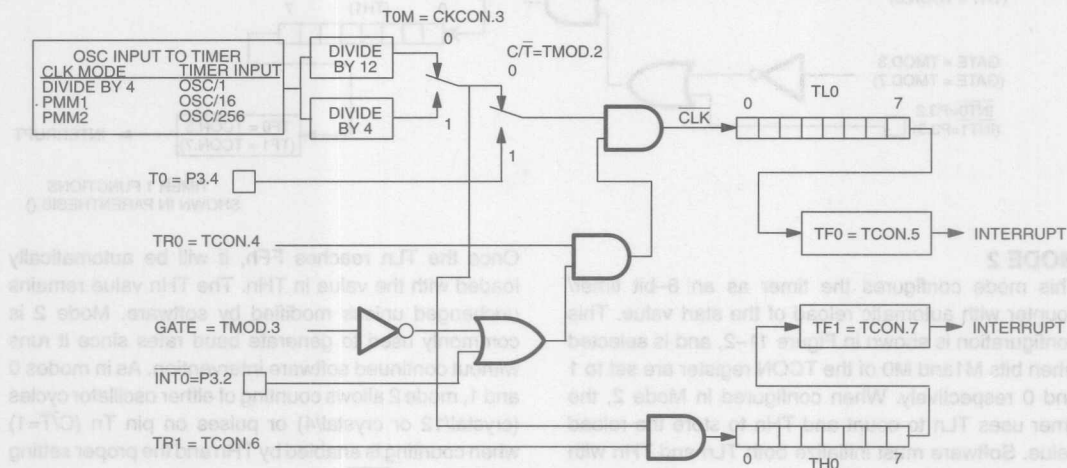
**TIMER/COUNTER 0 AND 1 MODE 2** Figure 11-2

**MODE 3**

This mode provides an 8-bit timer/counter and a second 8-bit timer as indicated in Figure 11-3. In Mode 3, TL0 is an 8-bit timer/counter controlled by the normal Timer 0 bits (TR0=TCN.4 and TF0=TCN.5). TL0 can be used to count oscillator cycles (crystal/12 or crystal/4) or 1 to 0 transitions on pin T0 as determined by C/T (TMOD.2). As in the other modes, the GATE function

can use  $\overline{\text{INT0}}$  to give external run control of the timer to an outside signal.

TH0 becomes an independent 8-bit Timer in Mode 3, however it can only count oscillator cycles (divided by 12 or 4) as shown in the figure. In this mode, some of Timer 1's control signals are used to manipulate TH0. That is, TR1 (TCN.6) and TF1 (TCN.7) become the relevant control and flag bits associated with TH0.

**TIMER/COUNTER 0 MODE 3 Figure 11-3**

In Mode 3, Timer 1 stops counting and holds its value. Thus Timer 1 has no practical application while in Mode 3.

As mentioned above, when Timer 0 is in Mode 3, it uses some of Timer 1's resources (i.e., TR1 and TF1). Timer 1 can still be used in Modes 0, 1, and 2 in this situation, but its flexibility becomes somewhat limited. While it maintains its basic functionality, its inputs and outputs are no longer available. Therefore when Timer 0 is in Mode 3, Timer 1 can only count oscillator cycles, and it does not have an interrupt or flag. With these limitations,

baud rate generation is its most practical application, but other time-base functions may be achieved by reading the registers.

**TIMER 2**

Like Timers 0 and 1, Timer 2 is a full function timer/counter, however it has several additional capabilities that make it more useful. Timer 2 has independent control registers in T2CON and T2MOD, and is based on count registers TL2 and TH2. All of these registers are described in detail below.

## T2CON REGISTER SUMMARY

## TIMER TWO CONTROL

## T2CON; C8h

T2CON.7

TF2 – Timer 2 Overflow Flag. Hardware will set TF2 when the Timer 2 overflows from FFFFh or from the count equal to the capture register in down count mode. It must be cleared to 0 by software. TF2 will only be set to a 1 if RCLK and TCLK are both cleared to a 0.

T2CON.6

EXF2 – Timer 2 External Flag. Hardware will set EXF2 when a reload or capture is caused by a falling transition on the T2EX pin (P1.1). EXEN2 must be set for this function. This flag must be cleared to 0 by software. Writing a one to this bit will force a timer interrupt if enabled.

T2CON.5

RCLK – Receive Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of received data in Serial Modes 1 or 3. RCLK=1 causes Timer 2 overflow to be used as the receive clock. RCLK=0 causes Timer 1 overflow to be used as the receive clock.

T2CON.4

TCLK – Transmit Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of Transmit data in Serial Modes 1 or 3. TCLK=1 causes Timer 2 overflow to be used as the transmit clock. TCLK=0 causes Timer 1 overflow to be used as the transmit clock.

T2CON.3

EXEN2 – Timer 2 External Enable. Setting this bit to a 1 allows a capture or reload to occur as a result of a falling transition on T2EX (P1.1), if Timer 2 is not generating baud rates for the serial port. EXEN2=0 causes Timer 2 to ignore all external events at T2EX.

T2CON.2

TR2 – Timer 2 run. Setting this bit to a 1 starts Timer 2. Setting it to a 0 stops Timer 2.

T2CON.1

C/T2 – Counter/Timer Select. Setting this bit to a 0 selects a timer function for Timer 2. Setting it to a 1 selects a counter of falling transitions on T2 (P1.0). Timer 2 runs at 4 clocks per tick or 12 clocks per tick as programmed by CKCON.5. Independent of this bit, Timer 2 also runs at 2 clocks per tick when used in either baud-rate generator or clock output mode.

T2CON.0

CP/RL2 – Capture/Reload Flag. When this bit is set to 1, Timer 2 captures will occur on 1 to 0 transitions of T2EX (P1.1) if EXEN2=1. When this bit is set to 0, auto-reloads will occur when Timer 2 overflows or when 1 to 0 transitions occur on T2EX if EXEN2=1. If either RCLK or TCLK is set to a 1 this bit will not function and the timer will function in an auto-reload mode following each overflow.



## TIMER TWO MODE CONTROL

T2MOD.7–2

T2MOD; C9h

Reserved

T2MOD.1

T2OE – Timer 2 Output Enable. Setting this bit to a 1 enables the Timer 2 to drive the T2 (P1.0) pin with a clock output. When T2OE=0, the T2 (P1.0) pin is used as either an input for Timer 2 or a standard port pin.

T2MOD.0

DCEN – Down Count Enable. When this bit is set to 1, the Timer 2 function counts up or down when in 16-bit auto-reload mode depending on T2EX (P1.1). When DCEN is set to a 0, the Timer 2 counts up only.

## TIMER 2 CAPTURE REGISTERS SUMMARY

### LEAST SIGNIFICANT BYTE CAPTURE OF TIMER 2

RCAP2L; CAh

RCAP2L.7–0

This register is used to capture the TL2 value when Timer 2 is configured in capture mode. RCAP2L is also used as the LSB of a 16-bit reload value when Timer 2 is configured in auto-reload mode.

### MOST SIGNIFICANT BYTE CAPTURE OF TIMER 2

RCAP2H; CBh

RCAP2H.7–0

This register is used to capture the TH2 value when Timer 2 is configured in capture mode. RCAP2H is also used as the MSB of a 16-bit reload value when Timer 2 is configured in auto-reload mode.

## TIMER 2 MODES

As is seen in the register descriptions, Timer 2 has several abilities not found in Timers 0 and 1. However, it does not offer the 13-bit and dual 8-bit modes. Thus it runs in 16-bit mode at all times. Also note that instead of offering an 8-bit auto-reload mode, Timer 2 has a 16-bit auto-reload mode. This mode uses the Timer Capture registers to hold the reload values. The modes available on Timer 2 are described below.

### 16-bit Timer/Counter

In this mode, Timer 2 performs a simple timer or counter function where it behaves similarly to mode 1 of Timers 0 and 1, but uses 16 instead of 8 bits. This mode, along with the optional capture mode described below, is illus-

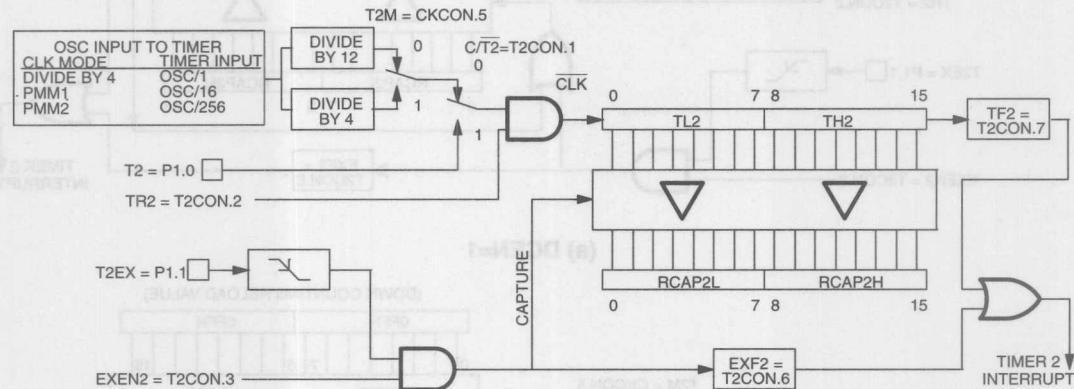
trated in Figure 11–4. The 16-bit count values are found in TL2 and TH2 Special Function Registers (addresses 0CCh and 0CDh respectively). The selection of whether a Timer or Counter function is performed is made using the bit  $C/\overline{T}2$  (T2CON.1). When  $C/\overline{T}2$  is set to a logic 1, Timer 2 behaves as a counter where it counts 1 to 0 transitions at the T2 (P1.0) pin. When  $C/\overline{T}2$  is set to a logic 0, Timer 2 functions as a timer where it counts the oscillator cycles divided by either 12 or 4 as determined by bit T2M (T2CON.5). Timing or counting is enabled by setting bit TR2 (T2CON.2) to 1, and disabled by setting it to 0. When the counter rolls over from FFFFh to 0000h, the TF2 flag (T2CON.7) is set and will cause an interrupt if Timer 2's interrupt is enabled.

### 16-bit Timer with Capture

A diagram of Timer 2's Capture Mode is shown in Figure 11-4. In this mode, the timer performs basically the same 16-bit timer/counter function described above. However, a 1 to 0 transition on T2EX (pin P1.1) causes the value in Timer 2 to be transferred into the capture registers if enabled by EXEN2 (T2CON.3). The capture registers, RCAP2L and RCAP2H, correspond to TL2

and TH2 respectively. The capture function is enabled by the CP/RL2 (T2CON.0) bit. When set to a logic 1, the timer is in capture mode as described. When set to a logic 0, the timer is in auto-reload mode described later. As was possible with Timers 0 and 1, the timebase for Timer 2 may be selected to be oscillator cycles divided by either 12 or 4 when in this mode.

**TIMER/COUNTER 2 WITH OPTIONAL CAPTURE** Figure 11-4

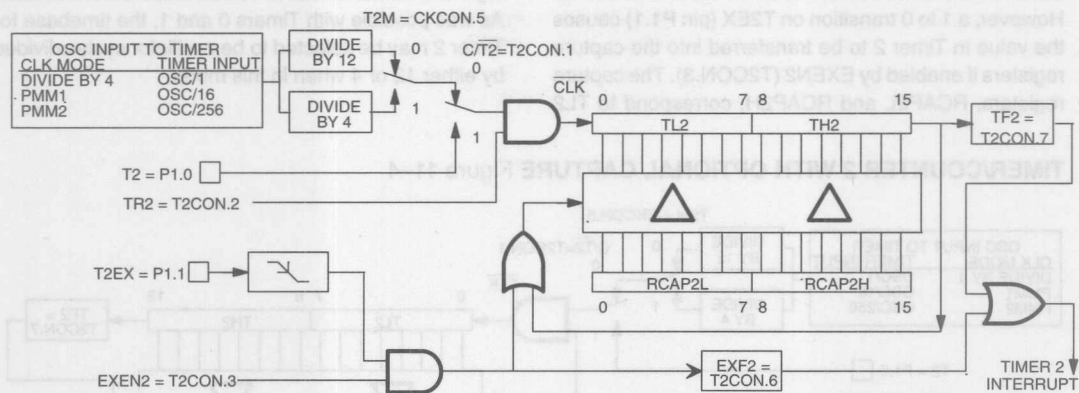
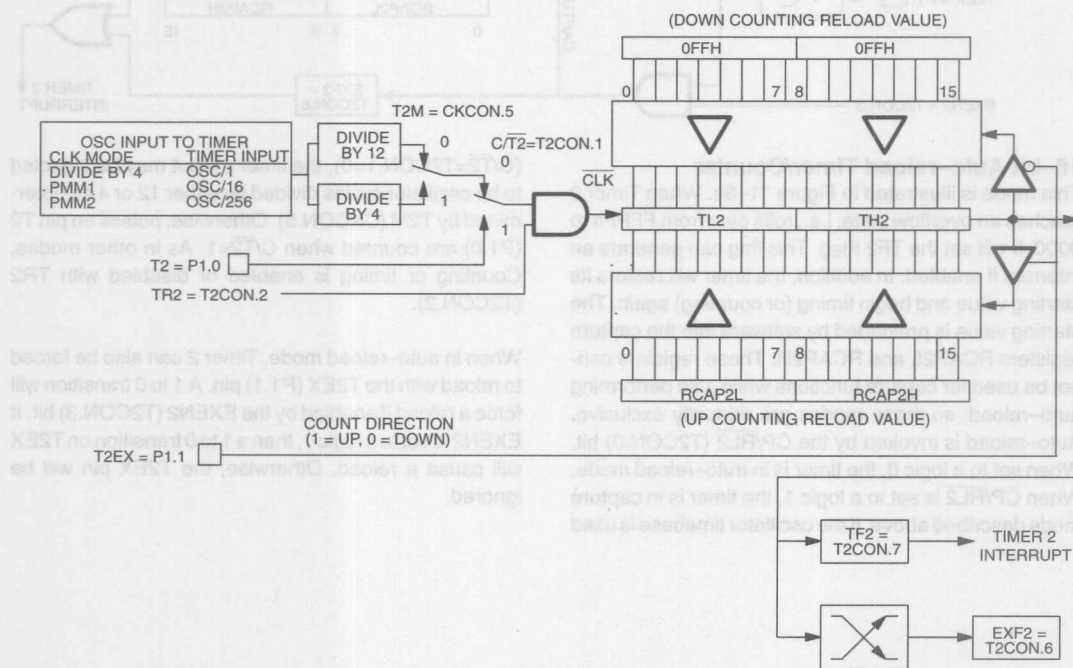


### 16-bit Auto-reload Timer/Counter

This mode is illustrated in Figure 11-5a. When Timer 2 reaches an overflow state, i.e., rolls over from FFFFh to 0000, it will set the TF2 Flag. This flag can generate an interrupt if enabled. In addition, the timer will restore its starting value and begin timing (or counting) again. The starting value is preloaded by software into the capture registers RCAP2L and RCAP2H. These registers cannot be used for capture functions while also performing auto-reload, so these modes are mutually exclusive. Auto-reload is invoked by the CP/RL2 (T2CON.0) bit. When set to a logic 0, the timer is in auto-reload mode. When CP/RL2 is set to a logic 1, the timer is in capture mode described above. If the oscillator timebase is used

( $C/\overline{T}2 = T2CON.1 = 0$ ), the timer's input may be selected to be oscillator cycles divided by either 12 or 4 as determined by T2M (CKCON.5). Otherwise, pulses on pin T2 (P1.0) are counted when  $C/\overline{T}2 = 1$ . As in other modes, Counting or timing is enabled or disabled with TR2 (T2CON.2).

When in auto-reload mode, Timer 2 can also be forced to reload with the T2EX (P1.1) pin. A 1 to 0 transition will force a reload if enabled by the EXEN2 (T2CON.3) bit. If EXEN2 is set to a logic 1, then a 1 to 0 transition on T2EX will cause a reload. Otherwise, the T2EX pin will be ignored.

**TIMER/COUNTER 2 AUTO RELOAD MODE** Figure 11-5**(a) DCEN=0****(a) DCEN=1**

### Up/Down Count Auto-reload Timer/Counter

When operating in auto-reload mode, Timer 2 can also function as an up/down auto-reload counter. This option is selected by the DCEN (T2MOD.0) bit, and is illustrated in Figure 11-5b. When DCEN is set to a logic 1, Timer 2 will count up or down as controlled by the state of pin T2EX (P1.1). T2EX will cause upward counting when a logic 1 is applied and down counting when a logic 0 is applied. When DCEN=0, Timer 2 only counts up.

When an overflow occurs in the upward counting direction, the value in RCAP2L and RCAP2H will be loaded into T2L and T2H respectively. In the down count direction, an underflow occurs when the values in T2L and T2H match the values in RCAP2L and RCAP2H respectively. When an underflow occurs, FFFFh is loaded into T2L and T2H and counting continues.

Note that in this mode, the overflow/underflow output of the timer is provided to an edge detection circuit as well as to the TF2 bit (T2CON.7). This edge detection circuit toggles the EXF2 bit (T2CON.6) on every overflow or underflow. Therefore, the EXF2 bit behaves as a seveneenth bit of the counter, and may be used as such.

### Baud Rate Generator

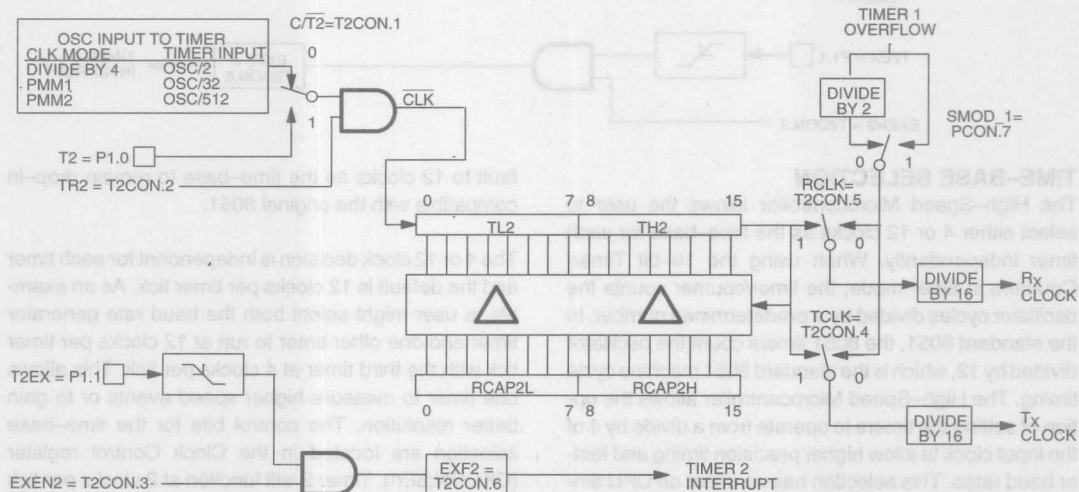
Timer 2 can be used to generate baud rates for Serial Port 0 in serial modes 1 or 3. Baud rate generator mode

is invoked by setting either the RCLK or TCLK bit in the T2CON register to a logic 1, as illustrated in Figure 11-6. In this mode, the timer continues to function in auto-reload mode, but instead of setting the interrupt flag T2F (T2CON.7) and potentially causing an interrupt, the overflow is used to generate the shift clock for the serial port function. As in normal auto-reload mode, an overflow causes the values previously loaded into RCAP2L and RCAP2H to be transferred into T2L and T2H respectively. Note that when RCLK or TCLK is set to 1, the Timer 2 is forced into 16-bit auto-reload mode regardless of the CP/RL2 bit.

As explained above, the timer itself cannot set the T2F interrupt flag and therefore cannot generate an interrupt. However if EXEN2 (T2CON.3) is set to 1, a 1 to 0 transition on the T2EX (P1.1) pin will cause the EXF2 (T2CON.6) interrupt flag to be set. If enabled, this will cause a Timer 2 Interrupt to occur. Therefore in this mode, the T2EX pin may be used as an additional external interrupt if desired.

Another somewhat unique feature of the baud rate generator mode is that the crystal derived timebase for the timer is the crystal frequency divided by 2. No other crystal divider selection is possible. If a different timebase is desired, bit C/T2 (T2CON.1) may be set to a 1 allowing the timebase to be derived from an external clock source supplied by the user on pin T2 (P1.0).

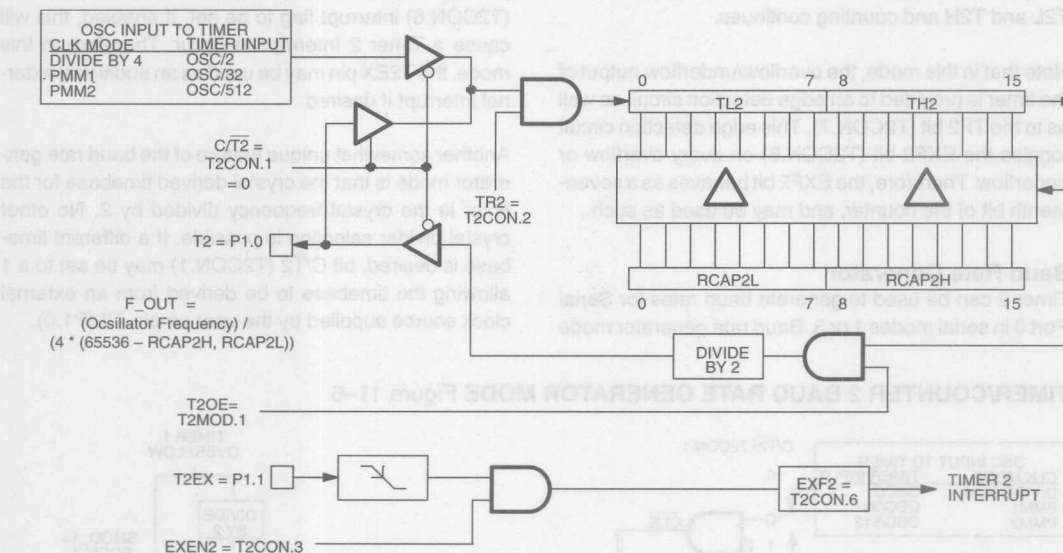
**TIMER/COUNTER 2 BAUD RATE GENERATOR MODE** Figure 11-6



Timer 2 can also be configured to drive a clock output on port pin P1.0 (T2) as shown in Figure 11–7. To configure Timer 2 for this mode, first it must be set to 16-bit auto-reload timer mode (CP/RL2=0, C/T2=0). Next, the T2OE (T2MOD.1) bit must be set to a logic 1. TR2 (T2CON.2) must also be set to a logic 1 to enable the timer.

This mode will produce a 50% duty cycle square wave output. The frequency of the square wave is given by the formula in the figure. Each timer overflow causes an edge transition on the pin, i.e., the state of the pin toggles.

**TIMER/COUNTER 2 CLOCK OUT MODE** Figure 11–7



### TIME-BASE SELECTION

The High-Speed Microcontroller allows the user to select either 4 or 12 clocks as the time-base for each timer independently. When using the 16-bit Timer/Counters in timer mode, the timer/counter counts the oscillator cycles divided by a predetermined number. In the standard 8051, the 8051 timers count the oscillator divided by 12, which is the standard 8051 machine cycle timing. The High-Speed Microcontroller allows the option of setting the timers to operate from a divide by 4 of the input clock to allow higher precision timing and faster baud rates. This selection has no effect on CPU timing, only on the timers. Following a reset, the timers de-

fault to 12 clocks as the time-base to remain drop-in compatible with the original 8051. First, the timebase is the crystal frequency divided by 2, and no other divider selection is possible. Second, the timer itself will not generate an interrupt, but if needed, an additional external interrupt may be caused using T2EX as described above. Because of the two mode's similarities, the timer can be used to generate both an external clock and a baud rate clock simultaneously. Once the clock out mode is established, either TCLK or RCLK is set to 1, and the RCAP2 registers are loaded, the timer will provide a clock to both functions.

fault to 12 clocks as the time-base to remain drop-in compatible with the original 8051.

The 4 or 12 clock decision is independent for each timer and the default is 12 clocks per timer tick. As an example, a user might select both the baud rate generator timer and one other timer to run at 12 clocks per timer tick with the third timer at 4 clocks per tick. This allows one timer to measure higher speed events or to gain better resolution. The control bits for the time-base selection are located in the Clock Control register (CKCON;8Eh). Timer 2 will function at 2 clocks per tick



when set for baud rate generation or clock output as described above.

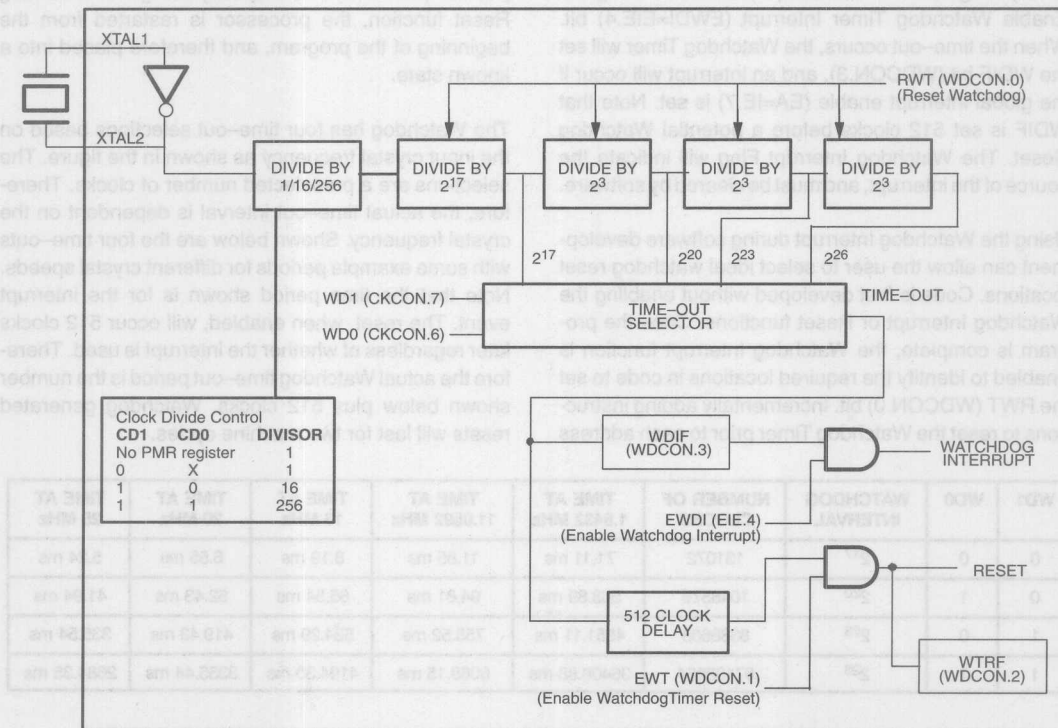
The use of Power Management modes will affect the input clock to the timer as shown in the illustrations. In general, they will divide the input clock by either 16 or 256 for PMM1 and PMM2, respectively. Timer 2, when operating in Baud Rate Generator or Clock Out mode normally uses the input clock frequency divided by 2, but when PMM1 and PMM2 are used, it will operate from a time-base of the input clock divided by 32 and 512, respectively.

### WATCHDOG TIMER

The Watchdog Timer is a user programmable clock counter that can serve as a time-base generator, an

event timer, or a system supervisor. As can be seen in the diagram of Figure 11–8, the timer is driven by the main system clock that is supplied to a series of dividers. The divider output is selectable, and determines the interval between time-outs. When the time-out is reached, an interrupt flag will be set, and if enabled, a reset will occur. The interrupt flag will cause an interrupt to occur if its individual enable bit is set and the global interrupt enable is set. The reset and interrupt are completely discrete functions that may be acknowledged or ignored, together or separately for various applications.

**WATCHDOG TIMER** Figure 11–8



The Watchdog Timer Reset function works as follows. After initializing the correct time-out interval (discussed below), software first restarts the Watchdog using RWT (WDCON.0) and then enables the reset mode by setting the Enable Watchdog Timer Reset (EWT=WDCON.1)

bit. At any time prior to reaching its user selected terminal value, software can set the Reset Watchdog Timer (RWT=WDCON.0) bit. If RWT is set before the time-out is reached, the timer will start over. If the time-out is reached without RWT being set, the Watchdog will reset

the CPU. Hardware will automatically clear RWT after software sets it. When the reset occurs, the Watchdog Timer Reset Flag (WTRF=WDCON.2) will automatically be set to indicate the cause of the reset, however software must clear this bit manually.

The Watchdog Timer is a free running timer. When used as a simple timer with both the reset and interrupt functions disabled (EWT=0 and EWDI=0), the timer will continue to set the Watchdog Interrupt flag each time the timer completes the selected timer interval as programmed by WD1 (CKCON.7) and WD2 (CKCON.6). Restarting the timer using the RWT (WDCON.0) bit, allows software to use the timer in a polled time-out mode. The WDIF bit is cleared by software or any reset.

The Watchdog Interrupt is also available for applications that do not need a true Watchdog Reset but simply a very long timer. The interrupt is enabled using the Enable Watchdog Timer Interrupt (EWDI=EIE.4) bit. When the time-out occurs, the Watchdog Timer will set the WDIF bit (WDCON.3), and an interrupt will occur if the global interrupt enable (EA=IE.7) is set. Note that WDIF is set 512 clocks before a potential Watchdog Reset. The Watchdog Interrupt Flag will indicate the source of the interrupt, and must be cleared by software.

Using the Watchdog Interrupt during software development can allow the user to select ideal watchdog reset locations. Code is first developed without enabling the Watchdog Interrupt or Reset functions. Once the program is complete, the Watchdog Interrupt function is enabled to identify the required locations in code to set the RWT (WDCON.0) bit. Incrementally adding instructions to reset the Watchdog Timer prior to each address

location (identified by the Watchdog Interrupt) will allow the code to eventually run without receiving a Watchdog Interrupt. At this point the Watchdog Timer Reset can be enabled without the potential of generating unwanted resets. At the same time the Watchdog Interrupt may also be disabled. Proper use of the Watchdog Interrupt with the Watchdog Reset allows interrupt software to survey the system for errant conditions.

When using the Watchdog Timer as a system monitor, the Watchdog Reset function should be used. If the Interrupt function were used, the purpose of the watchdog would be defeated. For example, assume the system is executing errant code prior to the Watchdog Interrupt. The interrupt would temporarily force the system back into control by vectoring the CPU to the interrupt service routine. Restarting the Watchdog and exiting by an RETI or RET, would return the processor to the lost position prior to the interrupt. By using the Watchdog Reset function, the processor is restarted from the beginning of the program, and therefore placed into a known state.

The Watchdog has four time-out selections based on the input crystal frequency as shown in the figure. The selections are a preselected number of clocks. Therefore, the actual time-out interval is dependent on the crystal frequency. Shown below are the four time-outs with some example periods for different crystal speeds. Note that the time period shown is for the interrupt event. The reset, when enabled, will occur 512 clocks later regardless of whether the interrupt is used. Therefore the actual Watchdog time-out period is the number shown below plus 512 clocks. Watchdog generated resets will last for two machine cycles.

WD1	WD0	WATCHDOG INTERVAL	NUMBER OF CLOCKS	TIME AT 1.8432 MHz	TIME AT 11.0592 MHz	TIME AT 16 MHz	TIME AT 20 MHz	TIME AT 25 MHz
0	0	$2^{17}$	131072	71.11 ms	11.85 ms	8.19 ms	6.55 ms	5.24 ms
0	1	$2^{20}$	1048576	568.89 ms	94.81 ms	65.54 ms	52.43 ms	41.94 ms
1	0	$2^{23}$	8388608	4551.11 ms	758.52 ms	524.29 ms	419.43 ms	335.54 ms
1	1	$2^{26}$	67108864	36408.88 ms	6068.15 ms	4194.30 ms	3355.44 ms	2684.35 ms

The Watchdog time-out selection is made using bits WD1 (CKCON.7) and WD0 (CKCON.6) as shown in the figure. The time-out selections possible are shown in the bit descriptions that follow. The watchdog timeout period is affected by the use of Power Management modes. The slower clock rate, either divide by 64 or divide by 1024 is used as the input source for the watchdog timer. This allows the watchdog period to remain synchronized with device operation.

As discussed above, the Watchdog Timer has several SFR bits that contribute to its operation. It can be enabled to function as either a reset source, interrupt source, software polled timer or any combination of the three. Both the reset and interrupt have status flags. The Watchdog also has a bit that restarts the timer. A summary table showing the bit locations is below. A description follows.

BIT NAME	DESCRIPTION	REGISTER LOCATION	BIT POSITION
EWT	Enable Watchdog Timer Reset	WDCON – D8h	WDCON.1
RWT	Reset Watchdog Timer	WDCON – D8h	WDCON.0
WD1	Watchdog interval 1	CKCON – 8Eh	CKCON.7
WD0	Watchdog interval 0	CKCON – 8Eh	CKCON.6
WTRF	Watchdog Timer Reset Flag	WDCON – D8h	WDCON.2
EWDI	Enable Watchdog Timer Interrupt	EIE – E8h	EIE.4
WDIF	Watchdog Interrupt Flag	WDCON – D8h	WDCON.3

The Watchdog Timer is a free running timer. It will be disabled by a Power-fail Reset. A Watchdog time-out reset will not disable the Watchdog Timer, but will restart the timer. In general, software should set the Watchdog

to whichever state is desired, just to be certain of its state. Control bits that support Watchdog operation are described below.

## WDCON REGISTER SUMMARY

### WATCHDOG CONTROL

#### WDCON; D8h

WDCON.3

WDIF – Watchdog Interrupt Flag. If the Watchdog Interrupt is enabled (EIE.4), hardware will set this bit to indicate that the Watchdog Interrupt has occurred. If the interrupt is not enabled, this bit indicates that the time-out has passed. If the Watchdog Reset is enabled (WDCON.1), the user has 512 clocks to strobe the Watchdog prior to a reset. Software or any reset can clear this flag.

WDCON.2

WTRF – Watchdog Timer Reset Flag. Hardware will set this bit when the Watchdog Timer causes a reset. Software can read it, but must clear it manually. A Power-fail Reset will also clear the bit. This bit assists software in determining the cause of a reset. If EWT=0, the Watchdog Timer will have no affect on this bit.

WDCON.1

EWT – Enable Watchdog Timer Reset. Setting this bit will turn on the Watchdog Timer Reset function. The interrupt will not occur unless the EWDI bit in the EIE register is set. A reset will occur according to the WD1 and WD0 bits in the CKCON register. Setting this bit to a 0 will disable the reset but leave the timer running.

WDCON.0

RWT – Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time-out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the

**CLOCK CONTROL****CKCON; 8EH**

CKCON.7

WD1 – Watchdog Timer mode select bit 1. See table below for operation.

CKCON.6

WD0 – Watchdog Timer mode select bit 0. See table below for operation.

The WD select bits determine the time-out period of the Watchdog Timer. The timer divides the crystal frequency by a programmable value as shown below. The divider value is expressed in number of clock (crystal) cycles. Note that the reset time-out is 512 clocks longer than the interrupt, regardless of whether the interrupt is enabled.			
WD1	WD0	Interrupt Divider	Reset Divider
0	0	$2^{17}$	$2^{17} + 512$
0	1	$2^{20}$	$2^{20} + 512$
1	0	$2^{23}$	$2^{23} + 512$
1	1	$2^{26}$	$2^{26} + 512$

The default Watchdog time-out is the shortest one (WD1=WD0=0). Software can change this value easily, so this should cause no inconvenience. However, the EWT, WDIF, and RWT bits are protected under the Timed Access procedure. This prevents software from

accidentally enabling or disabling the Watchdog. Most importantly, it prevents errant code from accidentally clearing and restarting the Watchdog. More details are discussed in the section on Timed Access.

## SECTION 12: SERIAL I/O

The High-Speed Microcontroller serial communication is compatible with the 80C32. This includes framing error detection and automatic address recognition. The High-Speed Microcontroller provides two fully independent UARTs (serial ports) for simultaneous communication over two channels. The UARTs can be operated in identical or different modes and communication speeds. In this documentation, all descriptions apply to both UARTs unless stated otherwise.

Each serial port is capable of both synchronous and asynchronous modes. In the synchronous mode, the microcontroller generates the clock and operates in a half-duplex mode. In the asynchronous mode, full duplex operation is available. Receive data is buffered in a holding register. This allows the UART to receive an incoming word before software has read the previous value. Each UART has an associated control register (SCON0, SCON1) and each has a transmit/receive register (SBUF0, SBUF1). The SFR locations are: SCON0, 98h; SBUF0, 99h; SCON1, C0h; SBUF1, C1h.

The SBUF location provides access to both transmit and receive registers. Reads are directed to the receive buffer and writes to the transmit buffer automatically.

### SERIAL MODE SUMMARY

Each port provides four operating modes. These offer different communication protocols and baud rates. These modes are summarized briefly as follows. Detailed descriptions are provided later in this section.

The use of Power management modes, if supported, will affect the internal clock rate and baud rate as shown in Table 7-4. The following descriptions assume that Power Management modes are not in use.

#### MODE 0

This mode provides synchronous communication with external devices. It is commonly used to communicate with serial peripherals. Serial I/O occurs on the RXD pin. The shift clock is provided on the TXD pin. Note that whether transmitting or receiving, the serial clock is generated by the High-Speed Microcontroller. Thus any device on the serial port in Mode 0 must accept the microcontroller as the master.

The baud rate in Mode 0 is a function of the oscillator input. It will be the clock input divided by either 12 or 4.

This is selected by the SM2 bit (SCON0.5 or SCON1.5) as described below. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic 1, the serial port runs at a divide by 4. With the exception of the additional new divide by 4 of the oscillator (supported by SM2), Mode 0 operation is identical to the 80C32.

#### MODE 1

This mode provides standard full duplex asynchronous communication. A total of 10 bits is transmitted including 1 start bit, 8 data bits, and 1 stop bit. The received stop bit is stored in bit location RB8 in the relevant SCON register.

In Mode 1, the baud rate is a function of timer overflow. This makes the baud rate programmable by the user. Mode 1 has a difference for the two UARTs. Serial Port 0 can use either Timer 1 or 2 to generate baud rates. Serial Port 1 can use only Timer 1. Note that if both serial ports use the same timer, they will be running at the same baud rate. If they use different timers (or different modes), they can run at different rates. Baud rates are discussed in more detail below. Mode 1 operation is identical to the standard 80C32 when Timers 1 or 2 use the default divide by 12 of the oscillator.

#### MODE 2

This mode is an asynchronous mode that transmits a total of 11 bits. These include 1 start bit, 8 data bits, a programmable ninth bit, and 1 stop bit. The ninth bit is determined by the value in TB8 (SCON0.3 or SCON1.3) for transmission. When the ninth bit is received, it is stored in RB8 (SCON0.2 or SCON1.2). The ninth bit can be a parity value by moving the P bit (PSW.0) to TB8.

The baud rate for Mode 2 is a function of the oscillator frequency. It is either the oscillator input divided by 32 or 64 as programmed by the SMOD bit in the PCON register. Mode 2 operation is identical to the standard 80C32.

#### MODE 3

This mode has the same functionality as Mode 2, but generates baud rates like Mode 1. That is, this mode transmits 11 bits, but generates baud rates via the timers. Like Mode 1, either Timer 1 or 2 can be used for Serial Port 0 and Timer 1 can be used for Serial Port 1. Mode 3 operation is identical to the standard 80C32 when Timers 1 or 2 use the default divide by 12 of the oscillator.



## SERIAL PORT INITIALIZATION

In order to use the UART function(s), the serial port must be initialized. This involves selecting the mode and time base, then initializing the baud rate generator if necessary. Serial communication is then available. Once the baud rate generator is running, the UART can receive data.

In Mode 0, the High-Speed Microcontroller must provide the clock. Serial reception is initiated by setting the RI bit to a logic 0 and REN to a logic 1. This will generate a clock on the TXD pin and shift in the 8 bits on the RXD

pin. In the other modes, setting the REN bit to a logic 1 will allow serial reception. The external device must actually initiate it by sending a start bit. In any mode, serial transmission is initiated by writing to either the SBUF0 or SBUF1 location.

Most of the serial port controls are provided by the SCON0 and SCON1 registers. For convenience, these are provided below. In addition, other control bits that influence the Serial Port operation are also summarized below.

## SERIAL I/O MODES

MODE	SYNCH/ASYNCH	BAUD CLOCK <sup>†</sup>	DATA BITS	START/STOP	9TH BIT FUNCTION
0	Synch	4 or 12 $t_{CLK}$	8	None	None
1	Asynch	Timer 1 or 2*	8	1 start, 1 stop	None
2	Asynch	32 or 64 $t_{CLK}$	9	1 start, 1 stop	0, 1, parity
3	Asynch	Timer 1 or 2*	9	1 start, 1 stop	0, 1, parity

\* Timer 2 available for Serial Port 0 only.

<sup>†</sup> The use of PMM1 or PMM2 will affect the baud clock.

## SERIAL PORT CONTROL ZERO

### SCON0; 98h

This is the standard 80C32 Serial Port. The new Serial Port is designated Serial Port 1 and is documented below.

#### SCON0.7

SM0/FE\_0 – Serial Port 0 Mode bit 0 or Framing Error Flag. PCON.6 (SMOD0) determines whether this bit functions as SM0 or FE. The operation of SM0 (SMOD0=0) is described in the table below. When SMOD0=1, the serial port will set FE to indicate an invalid stop bit. When used as FE, this bit must be cleared in software.

#### SCON0.6

SM1\_0 – Serial Port 0 mode select 1. The operation of SM1 is described in the table below.

#### SCON0.5

SM2\_0 – Multiple MCU communication. Setting this bit to a one enables multiprocessor communication in Modes 2 or 3. If the ninth bit is 0, the RI\_0 will not be set. In Mode 1, setting the SM2\_0 bit to a one causes the RI\_0 bit not to be set if a valid stop bit is not received. In the High-Speed Microcontroller, SM2\_0 also has a new function. In mode 0, the SM2\_0 bit controls whether the serial port clock runs at a divide by 4 or a divide by 12 of the oscillator when not in PMM. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic one, the serial port runs at a divide by 4. This results in much faster synchronous serial communication.

#### SCON0.4

REN\_0 – Receive Enable. When set to a 1, the receive shift register will be enabled.

- SCON0.3 TB8\_0 – Set/clear to define the state of the ninth transmission data bit in modes 2 and 3.
- SCON0.2 RB8\_0 – Indicates the state of an incoming ninth bit when in modes 2 and 3. In mode 1, when SM2=0, RB8\_0 is the state of the stop bit received. RB8\_0 is not used in mode 0.
- SCON0.1 TI\_0 – Flag that indicates the transmitted word has been completely shifted out. In mode 0, TI\_0 is set at the end of the eighth data bit. In all other modes, this bit is set at the end of the last data bit. It must be cleared manually by software.
- SCON0.0 RI\_0 – Flag that indicates a serial word has been received. In mode 0, RI\_0 is set at the end of the eighth bit. In mode 1, it is set after the last sample of the incoming stop bit subject to the state of SM2\_0. In modes 2 and 3, RI\_0 is set after the last sample of RB8\_0. It must be cleared manually by software.

SM0/FE_0	SM1_0	Mode	Function	Length	Period
0	0	0	Sync	8 bits	4/12 $t_{CLK}$ (see SM2)
0	1	1	Asynch	10 bits	Timer 1 or 2
1	0	2	Asynch	11 bits	64/32 $t_{CLK}$
1	1	3	Asynch	11 bits	Timer 1 or 2

Initialization: SCON is set to 00h on a reset.

Read/Write Access: Unrestricted.

#### SERIAL PORT CONTROL ONE

SCON1; C0h

Serial Port 1 performs identically to the standard Serial Port 0 on an 80C32 with one exception. The baud rate generation from Timer 2 is not available in Modes 1 and 3. Timer 1 is used. The port is located at P1.3 and P1.2 for TXD1 and RXD1 respectively.

- SCON1.7 SM0/FE\_1 – Serial Port 1 Mode bit 0 or Framing Error Flag. PCON.6 (SMOD0) determines whether this bit functions as SM0 or FE. The operation of SM0 (SMOD0=0) is described in the table below. When SMOD0=1, the serial port will set FE to indicate an invalid stop bit. When used as FE, this bit must be cleared in software.
- SCON1.6 SM1\_1 – Serial Port 1 mode select 1. The operation of SM1\_1 is described in the table below.
- SCON1.5 SM2\_1 – Multiple MCU communication. Setting this bit to a one enables multiprocessor communication in Modes 2 or 3. If the ninth bit is 0, the RI\_1 will not be set. In Mode 1, setting the SM2\_1 bit to a one causes the RI\_1 bit not to be set if a valid stop bit is not received. In the High-Speed Microcontroller, SM2\_1 also has a new function. In mode 0, the SM2\_1 bit controls whether the serial port clock runs at a divide by 4 or a divide by 12 of the oscillator when not in PMM. When set to a logic 0, the serial port runs at a divide by 12. When set to a logic one, the serial port runs at a divide by 4. This results in much faster synchronous serial communication.

SCON1.3 TB8\_1 – Set/clear to define the state of the ninth transmission data bit in modes 2 and 3.

SCON1.2 RB8\_1 – Indicates the state of an incoming ninth bit when in modes 2 and 3. In mode 1, when SM2=0, RB8 is the state of the stop bit received. RB8 is not used in mode 0.

SCON1.1 TI\_1 – Flag that indicates the transmitted word has been completely shifted out. In mode 0, TI is set at the end of the eighth data bit. In all other modes, this bit is set at the end of the last data bit. It must be cleared manually by software.

SCON1.0 RI\_1 – Flag that indicates a serial word has been received. In mode 0, RI\_1 is set at the end of the eighth bit. In mode 1, it is set after the last sample of the incoming stop bit subject to the state of SM2\_1. In modes 2 and 3, RI\_1 is set after the last sample of RB8\_1. It must be cleared manually by software.

SM0	SM1	Mode	Function	Length	Period
0	0	0	Sync	8 bits	4/12 t <sub>CLK</sub> (see SM2)
0	1	1	Asynch	10 bits	Timer 1
1	0	2	Asynch	11 bits	64/32 t <sub>CLK</sub>
1	1	3	Asynch	11 bits	Timer 1

Initialization: SCON1 is set to 00h on a reset.

Read/Write Access: Unrestricted.

## POWER CONTROL PCON; 87h

PCON.7 SMOD\_0. Doubles the serial baud rate in modes 1, 2, and 3 for Serial Port 0 (the standard port) when SMOD=1.

PCON.6 SMOD0 – Framing Error Detection Enable. When SMOD0 is set to a one, SCON0.7 and SCON1.7 are converted to the FE flag for the respective serial port. When SMOD0 is 0, then SCON0.7 and SCON1.7 are the SM0 function as defined for the serial port.

## WATCHDOG CONTROL WDCON; D8h

WDCON.7 SMOD\_1 – Serial Modification. When set to a logic 1, this bit doubles the baud rate of Serial Port 1. It works identically to PCON.7.

## TIMER TWO CONTROL T2CON; C8h

T2CON.5 RCLK – Receive Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of received data in Serial Modes 1 or 3. RCLK=1 causes Timer 2 overflow to be used as the receive clock. RCLK=0 causes Timer 1 overflow to be used as the receive clock.

T2CON.4 TCLK – Transmit Clock Flag. This bit determines whether Timer 1 or 2 is used for Serial Port 0 timing of Transmit data in Serial Modes 1 or 3. TCLK=1 causes Timer 2 overflow to be used as the transmit clock. TCLK=0 causes Timer 1 overflow to be used as the transmit clock.

## BAUD RATES

Each mode has a baud rate generator associated with it. This generator is generally the same for each UART. Several of the baud rate generation techniques have options and these options are independent for the two UARTs. The baud rate descriptions given below are separated by mode.

### Mode 0

Baud rates for this mode are driven directly from the crystal speed divided by either 12 or 4. Mode 0 is synchronous so that the shift clock output frequency will be the baud rate. The formula is simply as follows:

$$\text{Mode 0 baud rate} = \frac{\text{Oscillator Frequency}}{12}$$

$$\text{Mode 0 baud rate} = \frac{\text{Oscillator Frequency}}{4}$$

The default case is divide by 12. The user can select the divider using the SM2 bit in the associated SCON register. For Serial Port 0, the SM2\_0 bit is SCON0.5. For Serial Port 1, the SM2\_0 bit is SCON1.5. When SM2 is set to a logic 0, the baud rate is generated using a divide by 12 of the oscillator input. When SM2 is set to a logic 1, the baud rate is generated using divide by 4. Note that this use of SM2 differs from a standard 80C32. In that device, SM2 had no valid use when the UART was in Mode 0. Since it was generally set to a 0, for the divide by 12, there is no compatibility problem.

### Mode 2

In this asynchronous mode, baud rates are also generated from the oscillator input. This mode works identically to the original 8051 family. The baud rate is given by the following formula.

$$\text{Mode 2 baud rate} = \frac{2^{\text{SMOD}_x}}{64} * \text{Oscillator Frequency}$$

The result of this formula generates a baud rate of either  $1/32 * \text{oscillator frequency}$  or  $1/64 * \text{oscillator frequency}$ . In the formula, the numerator is expressed as two to the power of SMOD, where SMOD is either a 0 or 1. When SMOD=0, the numerator is a 1 and when SMOD=1, the numerator is a 2.

SMOD is a bit that effectively doubles the baud rate when set to a logic 1. For Serial Port 0, SMOD\_0 resides at PCON.7. This is the original location in the 8051 family. For Serial port 1, SMOD\_1 resides in WDCON.7. The SMOD bits are set to a logic 0 on reset, which gives the lower speed baud rate.

If the application determines that Mode 0 or 2 must be used, then the oscillator or crystal frequency must be selected to generate the correct baud rates since each mode offers two selections for a given frequency.

### Mode 1 or 3

These asynchronous modes are commonly used for communication with PCs, modems, and other similar interfaces. The baud rates are programmable using the oscillator input and 16-bit Timer 2 or 8-bit Timer 1. The respective timer is placed in auto-reload mode. Each time the timer reaches its rollover condition (FFFFh – Timer 2 or FFh – Timer 1), a clock is sent to the baud rate circuit. This clock is then divided by 16 to generate the exact baud rate. For Serial Port 0, either Timer 1 or 2 can be used to generate baud rates. Note that there are differences between the timers when used as baud rate generators. Serial Port 1 can use Timer 1 as a baud rate generator. Thus in Mode 1 or 3, the two serial ports can run at the same frequency if Timer 1 is used for both, but different frequencies if both timers are used.

Also note that the user can determine the speed at which Timer 1 runs (4 clocks or 12 clocks). In most cases, 12 clocks will be used for baud rate generation. Timer 2 runs from a 2 clock scheme when used for baud rate generation. This is compatible with the 80C32.

The baud rates for Mode 1 or 3 are given by these formulas.

### Serial Port 0 or 1

$$\text{Mode 1, 3 baud rate} = \frac{2^{\text{SMOD}_x}}{32} * \text{Timer 1 Overflow}$$

### Serial Port 0

$$\text{Mode 1, 3 baud rate} = \frac{\text{Timer 2 Overflow}}{16}$$

To use Timer 1 as the baud rate generator, it is commonly put into the 8-bit auto-reload mode. In this way,

the CPU is not involved in baud rate generation. Note that the timer interrupt should not be enabled. In the 8-bit auto-reload mode (Timer 1 Mode 2), the reload value is stored in TH1. Thus the combination of crystal frequency and TH1 determine the baud rate. The complete formula is as follows.

$$\text{Mode 1, 3 baud rate} = \frac{2^{\text{SMOD}_x} \cdot \text{Oscillator Frequency}}{32 \cdot 12 \cdot (256 - \text{TH1})}$$

Note that the 12 in the denominator can be changed to a 4 as determined by the Timer selection (T1M; CKCON.4). This formula provides the derived baud rate for a given TH1 and crystal. Most users already know what baud rate is desired and want the timer reload value. Thus the equation solves as follows, when T1M=0.

$$\text{TH1} = 256 - \frac{2^{\text{SMOD}_x} \cdot \text{Oscillator Frequency}}{384 \cdot \text{Baud Rate}}$$

Note that the most common application is to use Timer 1 in 8-bit auto-reload mode as a timer. It can actually be used in any mode and can also be configured as a counter.

To use Timer 2 as baud rate generator for Serial Port 0, the Timer is configured in auto-reload mode. Then either TCLK or RCLK bit (or both) are set to a logic 1. TCLK=1 selects Timer 2 as the baud rate generator for the transmitter and RCLK=1 selects Timer 2 for the receiver. Thus Serial Port 0 can have the transmit and receive operating at different baud rates by choosing Timer 1 for one data direction and Timer 2 for the other. Setting either RCLK or TCLK to a logic 1 selects Timer 2 for baud rate generation. RCLK and TCLK reside in T2CON.4 and TCON.5 respectively.

When using Timer 2 to generate baud rates, the formula will be as follows. Note that the reload value is a 16-bit number as compared with Timer 1, which uses only 8 bits.

$$\text{Mode 1, 3 baud rate} = \frac{\text{Oscillator Frequency}}{32 \cdot (65536 - \text{RCAP2H, RCAP2L})}$$

Note that the 32 in the denominator is a result of the timer being run at a divide by 2, combined with the divide by 16 applied to timer overflows as mentioned above. Timer 2 normally runs at a divide by either 12 or 4 in auto-reload mode. Setting RCLK or TCLK causes the divide by 2 operation.

This formula provides the derived baud rate for a given RCAP2H, RCAP2L and crystal. Most users already know what baud rate is desired and want the timer reload value. Thus the equation solves as follows.

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{Oscillator Frequency}}{32 \cdot \text{Baud Rate}}$$

The Timer 2 interrupt is automatically disabled when either RCLK or TCLK is set. Also, the TF2 (TCON.7) flag will not be set on a timer rollover. The manual reload pin, T2EX (P1.1), will not cause a reload either.

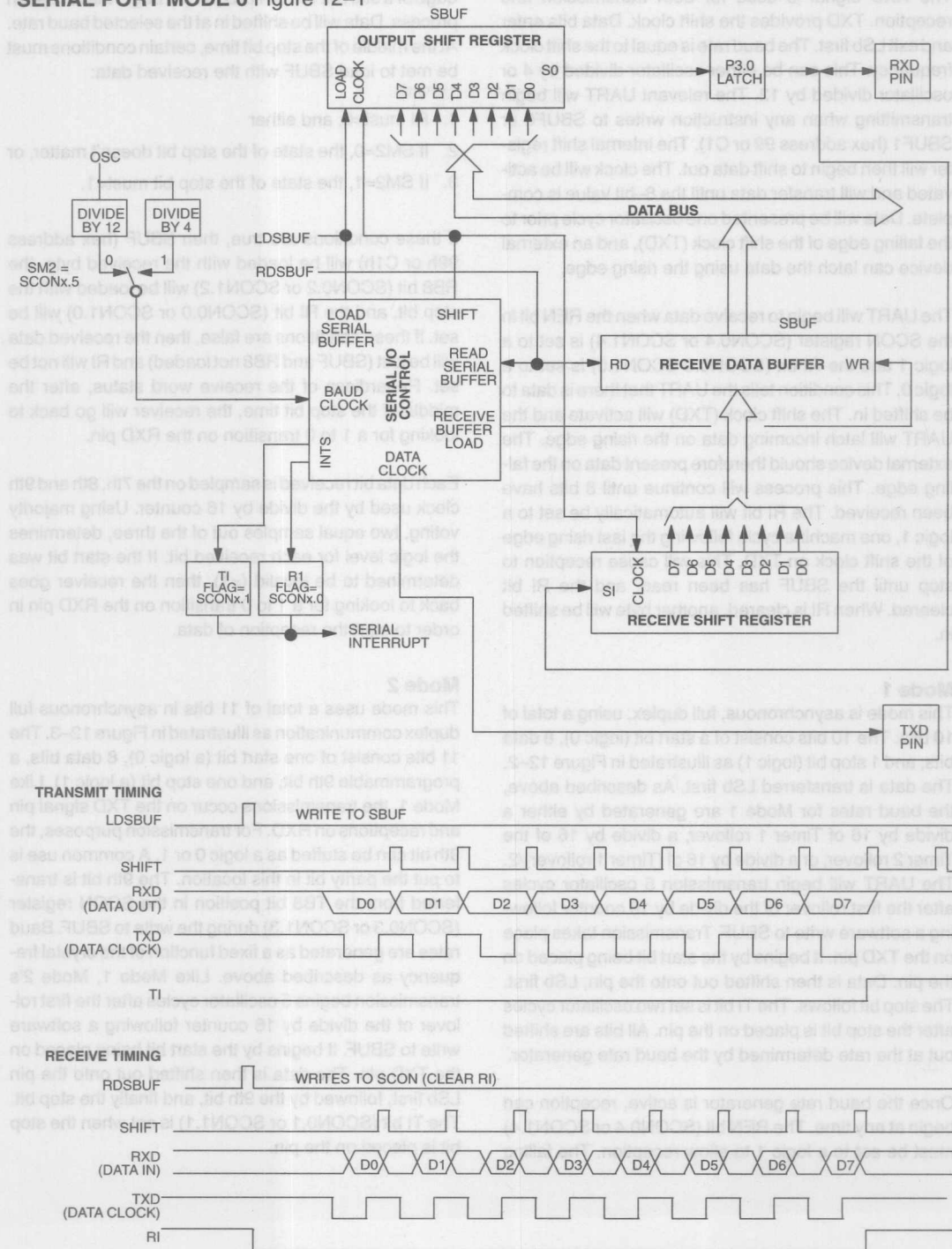
## SERIAL I/O DESCRIPTION

A detailed description of each serial mode is given below. A description of framing error detection and multiprocessor communication follows this section.

### Mode 0

This mode is used to communicate in synchronous, half-duplex format with devices that accept the High-Speed Microcontroller as a master. A functional block diagram and basic timing of this mode are shown in Figure 12-1. As can be seen, there is one bidirectional data line (RXD) and one shift clock line (TXD) used for communication. The shift clock is used to shift data into and out of the microcontroller and the remote device. Mode 0 requires that the microcontroller is the master because the microcontroller generates the serial shift clocks for both directions. As described above, the shift clock may be selected to be either divide by 12 or divide by 4 of the oscillator as determined by the SM2 (SCON0.5 or SCON1.5) bit.



**SERIAL PORT MODE 0 Figure 12-1**

The RXD signal is used for both transmission and reception. TXD provides the shift clock. Data bits enter and exit LSb first. The baud rate is equal to the shift clock frequency. This can be either oscillator divided by 4 or oscillator divided by 12. The relevant UART will begin transmitting when any instruction writes to SBUF0 or SBUF1 (hex address 99 or C1). The internal shift register will then begin to shift data out. The clock will be activated and will transfer data until the 8-bit value is complete. Data will be presented one oscillator cycle prior to the falling edge of the shift clock (TXD), and an external device can latch the data using the rising edge.

The UART will begin to receive data when the REN bit in the SCON register (SCON0.4 or SCON1.4) is set to a logic 1 and the RI bit (SCON0.0 SCON1.0) is set to a logic 0. This condition tells the UART that there is data to be shifted in. The shift clock (TXD) will activate and the UART will latch incoming data on the rising edge. The external device should therefore present data on the falling edge. This process will continue until 8 bits have been received. The RI bit will automatically be set to a logic 1, one machine cycle following the last rising edge of the shift clock on TXD. This will cause reception to stop until the SBUF has been read, and the RI bit cleared. When RI is cleared, another byte will be shifted in.

### Mode 1

This mode is asynchronous, full duplex, using a total of 10 bits. The 10 bits consist of a start bit (logic 0), 8 data bits, and 1 stop bit (logic 1) as illustrated in Figure 12-2. The data is transferred LSb first. As described above, the baud rates for Mode 1 are generated by either a divide by 16 of Timer 1 rollover, a divide by 16 of the Timer 2 rollover, or a divide by 16 of (Timer 1 rollover)/2. The UART will begin transmission 5 oscillator cycles after the first rollover of the divide by 16 counter following a software write to SBUF. Transmission takes place on the TXD pin. It begins by the start bit being placed on the pin. Data is then shifted out onto the pin, LSb first. The stop bit follows. The TI bit is set two oscillator cycles after the stop bit is placed on the pin. All bits are shifted out at the rate determined by the baud rate generator.

Once the baud rate generator is active, reception can begin at any time. The REN bit (SCON0.4 or SCON1.4) must be set to a logic 1 to allow reception. The falling

edge of a start bit on the RXD pin will begin the reception process. Data will be shifted in at the selected baud rate. At the middle of the stop bit time, certain conditions must be met to load SBUF with the received data:

1. RI must=0, and either
2. If SM2=0, the state of the stop bit doesn't matter, or
3. If SM2=1, the state of the stop bit must=1.

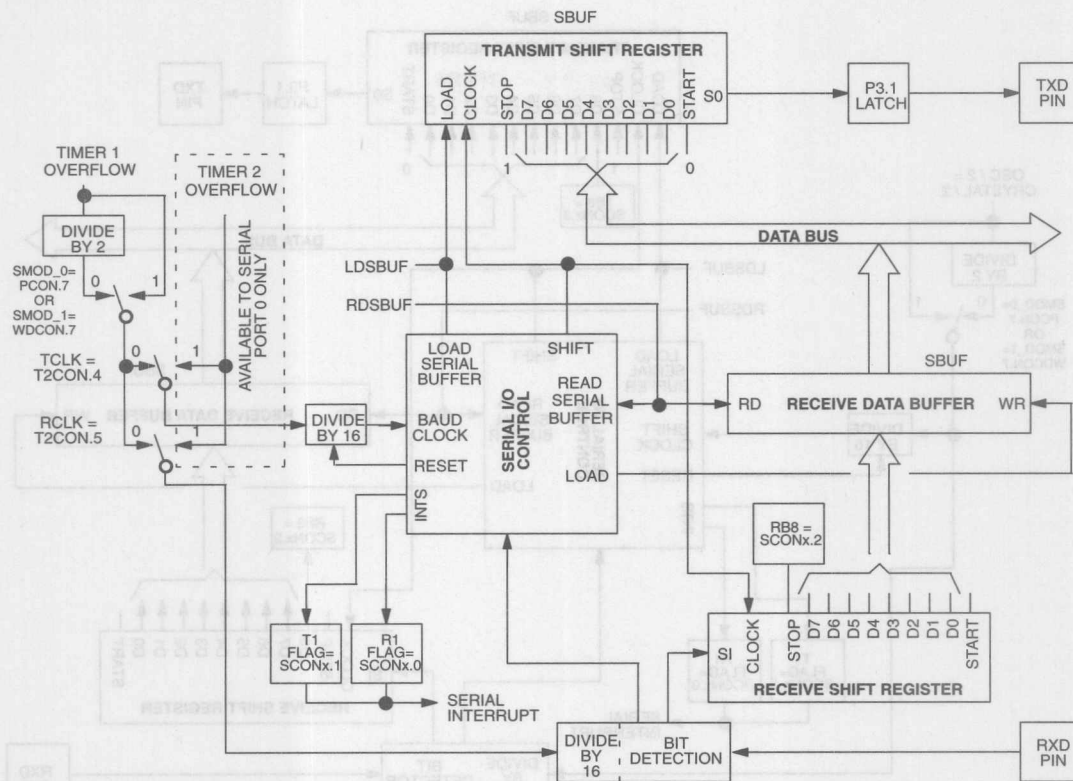
If these conditions are true, then SBUF (hex address 99h or C1h) will be loaded with the received byte, the RB8 bit (SCON0.2 or SCON1.2) will be loaded with the stop bit, and the RI bit (SCON0.0 or SCON1.0) will be set. If these conditions are false, then the received data will be lost (SBUF and RB8 not loaded) and RI will not be set. Regardless of the receive word status, after the middle of the stop bit time, the receiver will go back to looking for a 1 to 0 transition on the RXD pin.

Each data bit received is sampled on the 7th, 8th and 9th clock used by the divide by 16 counter. Using majority voting, two equal samples out of the three, determines the logic level for each received bit. If the start bit was determined to be invalid (=1), then the receiver goes back to looking for a 1 to 0 transition on the RXD pin in order to start the reception of data.

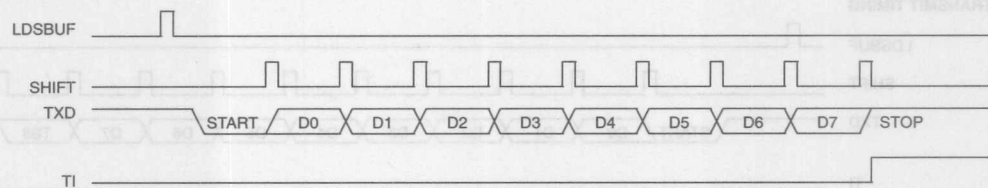
### Mode 2

This mode uses a total of 11 bits in asynchronous full duplex communication as illustrated in Figure 12-3. The 11 bits consist of one start bit (a logic 0), 8 data bits, a programmable 9th bit, and one stop bit (a logic 1). Like Mode 1, the transmissions occur on the TXD signal pin and receptions on RXD. For transmission purposes, the 9th bit can be stuffed as a logic 0 or 1. A common use is to put the parity bit in this location. The 9th bit is transferred from the TB8 bit position in the SCON register (SCON0.3 or SCON1.3) during the write to SBUF. Baud rates are generated as a fixed function of the crystal frequency as described above. Like Mode 1, Mode 2's transmission begins 5 oscillator cycles after the first rollover of the divide by 16 counter following a software write to SBUF. It begins by the start bit being placed on the TXD pin. The data is then shifted out onto the pin LSb first, followed by the 9th bit, and finally the stop bit. The TI bit (SCON0.1 or SCON1.1) is set when the stop bit is placed on the pin.

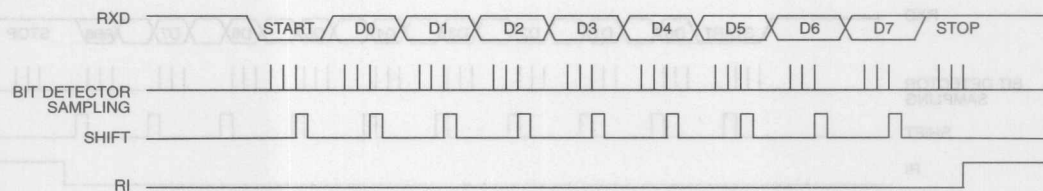
### SERIAL PORT MODE 1 Figure 12-2

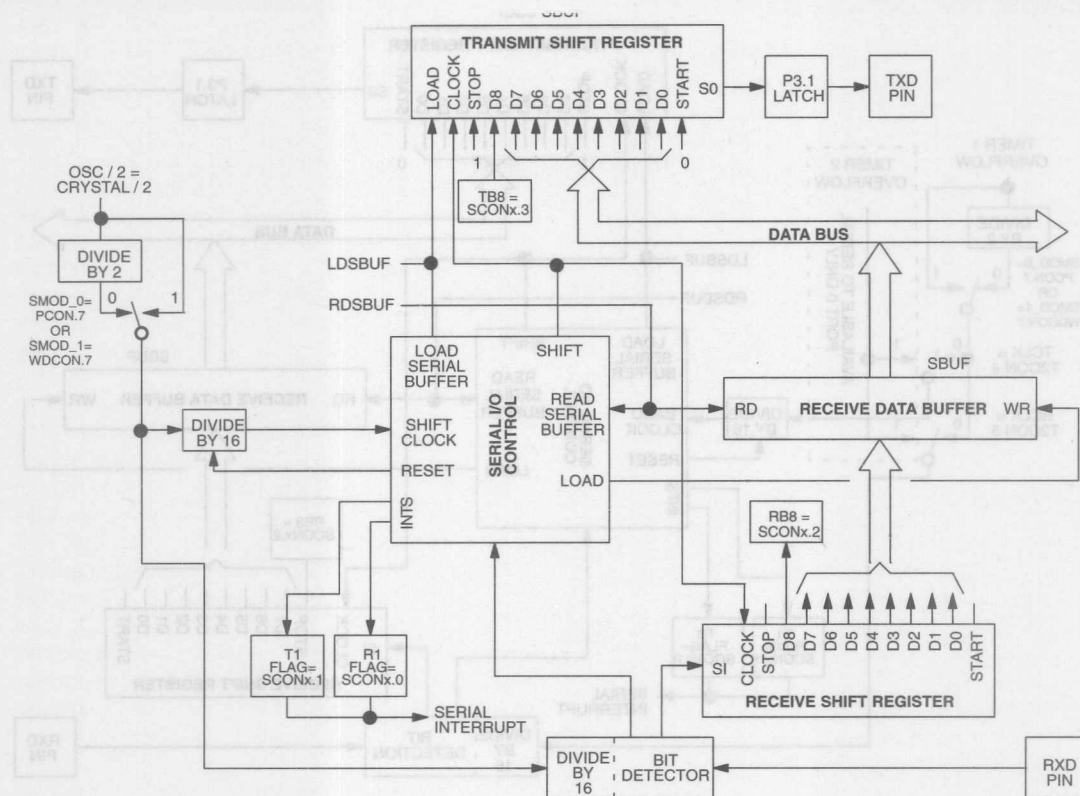


## TRANSMIT TIMING

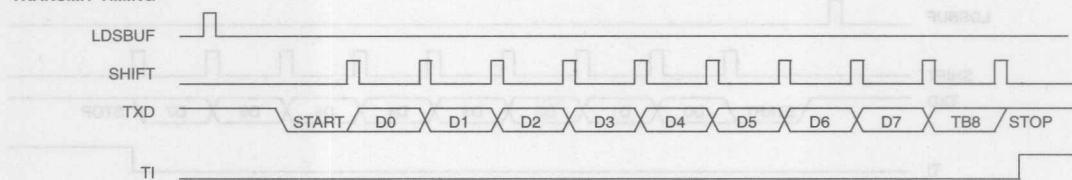


## RECEIVE TIMING

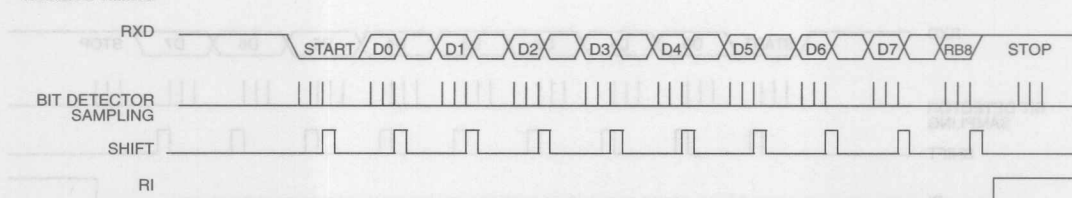




#### TRANSMIT TIMING



#### RECEIVE TIMING



Reception begins when a falling edge is detected as part of the incoming start bit on the RXD pin. The RXD pin is then sampled according to the baud rate speed. The 9th bit is placed in the RB8 bit location in SCON (SCON0.2 or SCON1.2). When a stop bit has been received, the data value will be transferred to the SBUF receive register (hex address 99 or C1). The RI bit (SCON0.0 or SCON1.0) will be set to indicate that a byte has been received. At this time, the UART can receive another byte.

Once the baud rate generator is active, reception can begin at any time. The REN bit (SCON0.4 or SCON1.4) must be set to a logic 1 to allow reception. The falling edge of a start bit on the RXD pin will begin the reception process. Data must be shifted in at the selected baud rate. At the middle of the 9th bit time, certain conditions must be met to load SBUF with the received data.

1. RI must=0, and either
2. If SM2=0, the state of the 9th bit doesn't matter, or
3. If SM2=1, the state of the 9th bit must=1.

If these conditions are true, then SBUF will be loaded with the received byte, RB8 will be loaded with the 9th bit, and RI will be set. If these conditions are false, then the received data will be lost (SBUF and RB8 not loaded) and RI will not be set. Regardless of the receive word status, after the middle of the stop bit time, the receiver will go back to looking for a 1 to 0 transition on RXD.

Data is sampled in a similar fashion to Mode 1 with the majority voting on three consecutive samples. Mode 2 uses the sample divide by 16 counter with either the oscillator divided by 2 or 4.

### Mode3

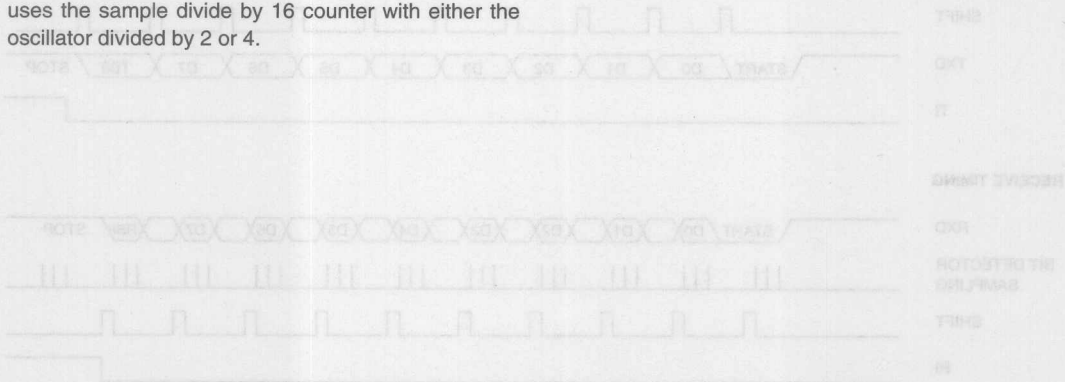
This mode has the same operation as Mode 2, except for the baud rate source. As shown in Figure 12-4, Mode 3 can use Timer 1 or 2 for Serial Port 0 and Timer 1 for Serial Port 1. The bit shifting and protocol are the same.

### FRAMING ERROR DETECTION

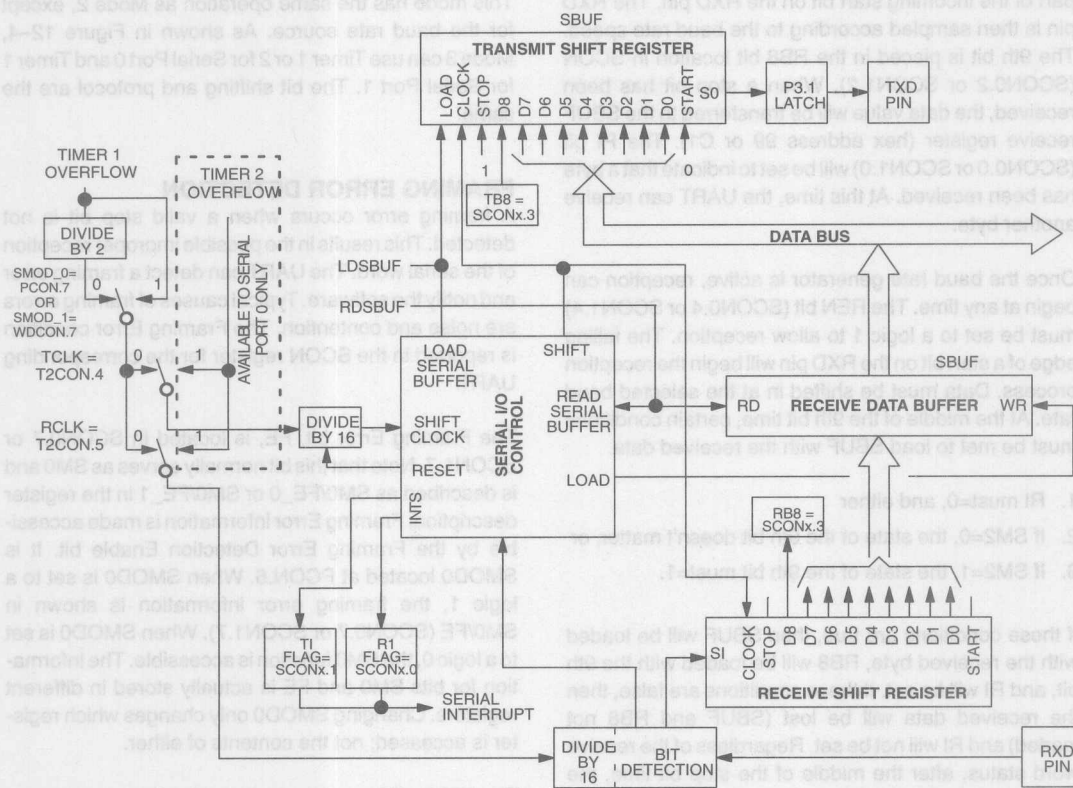
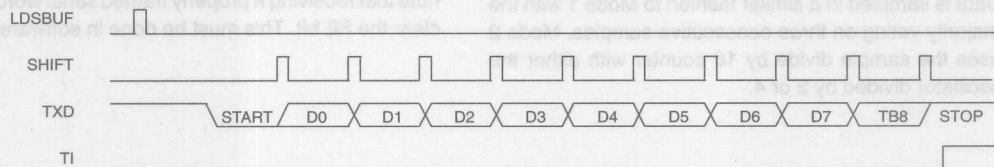
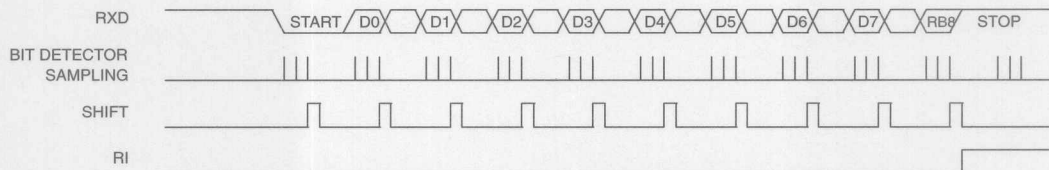
A framing error occurs when a valid stop bit is not detected. This results in the possible improper reception of the serial word. The UART can detect a framing error and notify the software. Typical causes of framing errors are noise and contention. The Framing Error condition is reported in the SCON register for the corresponding UART.

The Framing Error bit, FE, is located in SCON0.7 or SCON1.7. Note that this bit normally serves as SM0 and is described as SM0/FE\_0 or SM0/FE\_1 in the register description. Framing Error information is made accessible by the Framing Error Detection Enable bit. It is SMOD0 located at PCON.6. When SMOD0 is set to a logic 1, the framing error information is shown in SM0/FE (SCON0.7 or SCON1.7). When SMOD0 is set to a logic 0, the SM0 function is accessible. The information for bits SM0 and FE is actually stored in different registers. Changing SMOD0 only changes which register is accessed; not the contents of either.

The FE bit will be set to a 1 when a framing error occurs. It must be cleared by software. Note that the SMOD0 state must be 1 while reading or writing the FE bit. Also note that receiving a properly framed serial word will not clear the FE bit. This must be done in software.





**SERIAL PORT MODE 3 Figure 12-4****TRANSMIT TIMING****RECEIVE TIMING**

## MULTIPROCESSOR COMMUNICATION

Multiprocessor communication mode makes special use of the 9th data bit in Modes 2 and 3. In the original 8051, the 9th bit was restricted to a 0 or 1 condition, but had no special purpose. In the 80C32 and the High-Speed Microcontroller, it can be used to signify that the incoming byte is an address. This allows the processor to be interrupted only if the correct address appears. The Receive Interrupt, if enabled, will only occur when a recognized address is received.

When a serial word is received with the 9th bit set, the byte will be assumed to be an address. The address will be compared to an internally stored address. If it matches, a receive interrupt will occur. The internal address is derived from the contents of two registers. The first register specifies an absolute address. This is the user specified address of the device. The second register tells the comparator which address bit(s) to actually use in the comparison. This allows broadcast transmissions that reach groups of microcontrollers or all microcontrollers on a serial port. The user defines this protocol.

There are two Special Function Registers that support multiprocessor communication for each UART. These are independent, so that different addresses can be used in each. The registers are SADDR0 or SADDR1 (hex address A9 or A10) and SADEN0 or SADEN1 (hex address B9 or B10). The SADDR register specifies the individual processor's address. The SADEN identifies address bits that should be ignored in matching addresses.

Software will write an 8-bit address to the SADDR register. This is the microcontroller's individual address. Any bit in SADEN that contains a logic 0 will cause the corresponding bit in SADDR to be ignored in comparison. Thus logic 0 bits in SADEN create don't care bit states for address comparisons.

When an address is received, each address bit that is not masked by a don't care will be compared to the SADDR. The microcontroller will interrupt on any

address that matches this comparison. Any address that meets this comparison is called a Given Address.

The following example shows how one address can be directed to an individual processor, or two out of three.

### Micro 1

SADDR	11110000
SADEN	11111010
-----	
Given	11110x0x

### Micro 2

SADDR	11110001
SADEN	11111001
-----	
Given	11110xx1

### Micro 3

SADDR	11110010
SADEN	11111010
-----	
Given	11110x1x

Note that an address of 11110000 will reach only microcontroller 1. An address of 11110001 will reach both microcontroller 1 and microcontroller 2. An address of 11110010 will reach only microcontroller 3.

The microcontroller will also match on any address that corresponds to the Broadcast Address. This is the logical OR of the SADDR and SADEN registers, with any 0s defined as don't cares. In most cases, the Broadcast Address will be FFh.

The multiprocessor communication is always enabled. However, the SADEN registers default to 00h, which means all address bits are don't care, so all match. Thus if no multiprocessor communication is used, these registers can be ignored.

The Timer 2 interrupt is automatically disabled when either RCLK or TCLK is set. Also, the TF2 (TCON.7) flag will not be set on a timer rollover. The manual reload pin, T2EX (P1.1), will not cause a reload either.

ture called Timed Access to prevent accidental writes to critical SFR bits. These bits could cause a system failure or prevent the Watchdog Timer from doing its job if improperly written. The Timed Access involves opening a timing window during which the protected bit can be modified. If the window is opened correctly, it remains open long enough to alter one protected bit. This section explains which bits are protected, why, and how to use the Timed Access feature.

## PROTECTED BITS

Bits which are protected by the Timed Access feature are shown below. Only critical function bits which are unique to the High-Speed Microcontroller family are protected, assuring code compatibility with the original 80C51 or 80C52. A full description of the function of each bit is provided in Section 4.

EXIF.0	BGS	Band-gap Select
WDCON.6	POR	Power-on Reset Flag
WDCON.1	EWT	Watchdog Reset Enable
WDCON.0	RWT	Reset Watchdog Timer
WDCON.3	WDIF	Watchdog Interrupt Flag
TRIM.7	E4K	4096 Hz RTC Output
TRIM.6	X12/6	12pF/6pF Crystal Select
TRIM.5	TRM2	Capacitance Trim Bit 2
TRIM.4	TRM2	Inverse Capacitance Trim Bit 2
TRIM.3	TRM1	Capacitance Trim Bit 1
TRIM.2	TRM1	Inverse Capacitance Trim Bit 1
TRIM.1	TRM0	Capacitance Trim Bit 0
TRIM.0	TRM0	Inverse Capacitance Trim Bit 0
ROMSIZE.2	RMS2	ROM Size Select Bit 2
ROMSIZE.1	RMS1	ROM Size Select Bit 1

RTCC.0 RTCE RTC Enable

## PROTECTION SCHEME

Each bit mentioned above is protected against an accidental write by requiring the software to perform a procedure before writing the bit. Timed Access requires the software to write two specific values to the Timed Access register during two consecutive instruction cycles. The values AAh, then 55h, must be written in consecutive instructions to the TA register at SFR location C7h. If the writes are performed correctly, the write access window will open for three machine cycles. During this window, the software may modify a protected bit. The suggested code to open a Timed Access window is:

```
MOV 0C7h, #0AAh
MOV 0C7h, #55h
```

The procedure to modify a Timed Access protected bit begins by writing the value AAh to the Time Access register (TA;C7h). The value 55h must then be written to the Timed Access register within three machine cycles of writing AAh. This opens a three machine cycle window, after the write of 55h, during which any Timed Access protected bits may be modified. Failure to complete any of the required steps will also require the procedure to begin again, starting with the write of AAh to the Timed Access register. Attempts to modify Timed Access protected bits after the window has closed will be ignored. This is regardless of whether any bits were modified. Figure 13-1 illustrates a number of examples of correct and incorrect use of the Timed Access procedure.

**TIMED ACCESS EXAMPLES** Figure 13–1

three machine cycles MOV 0C7h, #0AAh	three machine cycles MOV 0C7h, #55h	two machine cycles SETB EWT	
three machine cycles MOV 0C7h, #0AAh	three machine cycles MOV 0C7h, #55h	one machine cycle NOP	two machine cycles SETB EWT
three machine cycles MOV 0C7h, #0AAh	three machine cycles MOV 0C7h, #55h	three machine cycles MOV WDCON, #02h	

**VALID TIMED ACCESS PROCEDURES**

three machine cycles MOV 0C7h, #0AAh	one machine cycle NOP	three machine cycles MOV 0C7h, #55h	two machine cycles SETB EWT
---	--------------------------	--	--------------------------------

\*Second write to TA register does not occur within 3 cycles of first write.

three machine cycles MOV 0C7h, #0AAh	three machine cycles MOV 0C7h, #55h	one machine cycle NOP	three machine cycles MOV WDCON, #02h
---	--	--------------------------	---

\*Modification of protected bit did not occur with 3 cycles of second write to TA register.

three machine cycles MOV 0C7h, #0AAh	three machine cycles MOV 0C7h, #55h	two machine cycles SETB EWT	two machine cycles SETB RWT
---	--	--------------------------------	--------------------------------

\*Modification of second protected bit did not complete within 3 cycles of second write to TA register.

**INVALID TIMED ACCESS PROCEDURES****TIMED ACCESS PROTECTS WATCHDOG**

Any microcontroller-based system can be faced with environmental conditions that are beyond its designed abilities. These include external signal transients due to component failure, fluctuating power conditions, massive electrostatic discharge (ESD), and other unexpected system events. When a microcontroller is exposed to such conditions, program execution can become corrupted. Members of the High-Speed Microcontroller family which incorporate a Watchdog Timer can initiate a reset to recover from these conditions. The primary function of the Timed Access feature is to protect against accidental disabling of the watchdog timer by an "out-of-control" device. This will allow the watch-

dog timer to reset the system in the event of program execution failure.

The following hypothetical example demonstrates how a single bit change can corrupt program execution. The Timed Access procedure protects against an accidental write to the EWT bit by the errant code, allowing the watchdog timer reset function to reset the device. While this is a purely fictitious example, it illustrates how the watchdog timer and Timed Access feature make the High-Speed Microcontroller minimize the effect of accidental code corruption. *Note: Timed Access is not optional and must be supported if the protected bits are used. This example simply helps explain the category of problem that the Timed Access prevents.*

## EXAMPLE: A TRANSIENT CAUSES THE WATCHDOG TO BE DISABLED

```

TABLE_READ:
C2D2  90 0A 00      MOV      DPTR, 0A00H      ;LOAD TABLE POINTER
C2D5  79 FF          MOV      R1, #0FFH        ;LOAD COUNTER
C2D7  78 90          MOV      R0, #90H         ;DESTINATION POINTER

      LOOP:
C2D9  E0             MOVX     A, @DPTR          ;READ DATA BYTE
C2DA  F6             MOV      @R0, A           ;STORE IT IN RAM
C2DB  06             INC      R0               ;NEXT TABLE LOCATION
C2DC  A3             INC      DPTR             ;NEXT DATA VALUE
C2DD  D9 C2 D9       DJNZ     R1, LOOP          ;NEXT BYTE OR DONE ?

```

A transient occurs while the opcode is being fetched for the first instruction. The transient causes one bit of the opcode in the first instruction to be read as a 0 instead of

1. The resulting program is what the microcontroller would actually execute:

```

TABLE_READ:
C2D2  80 0A 00      SJMP      0BH              ;RELATIVE JUMP BY 10 LOCATIONS
C2D5  79 FF          MOV      R1, #0FFH        ;LOAD COUNTER
C2D7  78 90          MOV      R0, #90H         ;DESTINATION POINTER

      LOOP:
C2D9  E0             MOVX     A, @DPTR          ;READ DATA BYTE
C2DA  F6             MOV      @R0, A           ;STORE IT IN RAM
C2DB  06             INC      R0               ;NEXT TABLE LOCATION
C2DC  A3             INC      DPTR             ;NEXT DATA VALUE
C2DD  D9 C2 D9       DJNZ     R1, LOOP          ;NEXT BYTE OR DONE ?

```

The resulting jump is to address C2DE. This is not even a real opcode, but would be treated as such. The resulting fetch is the value C2 D9. This is the opcode for CLR D9h. The bit addressable location D9h corresponds to the EWT – Enable Watchdog Timer. If the Timed Access procedure did not prevent it, this errant instruction would disable the Watchdog. Note that now, the program execution is completely lost. Real opcodes are being replaced by operands, data and garbage. In the High-Speed Microcontroller, the Watchdog will recover from

this state as soon as it times out since it could not have been disabled in this way.

In the High-Speed Microcontroller it is very hard to contrive a situation that will accidentally disable the Watchdog. Note, the Timed Access prevents accidentally writing a bit. It can not prevent accidentally calling the correct code that writes a bit. This is much more unlikely however.



**SECTION 14: REAL-TIME CLOCK**

The DS87C530 incorporates a real-time clock (RTC) onto the High-Speed Microcontroller family core. This allows the device to perform real-time related functions such as data logging and time-stamping without an external timer. In addition, the RTC includes an alarm func-

tion which can execute a software interrupt or resume operation from Stop mode at a specified time. The RTC features are controlled by 12 new SFRs. These registers, as well as two new interrupt control bits are shown in Table 14-1.

**REAL-TIME CLOCK CONTROL AND STATUS BIT SUMMARY** Table 14-1

BIT NAME	LOCATION	FUNCTION	RANGE	RESET	READ/WRITE ACCESS
ERTCI	EIE.5	RTC Interrupt Enable		0	Unrestricted
PRTCI	EIP.5	RTC Interrupt Priority		0	Unrestricted
RTASS.7-0	RTASS	RTC Alarm Subsecond	0-FFh	Unchanged	Unrestricted
RTAS.5-0	RTAS	RTC Alarm Second	0-3Bh	Unchanged	Unrestricted
RTAM.5-0	RTAM	RTC Alarm Minute	0-3Bh	Unchanged	Unrestricted
RTAH.4-0	RTAH	RTC Alarm Hour	0-17H	Unchanged	Unrestricted
RTCSS.7-0	RTCSS	RTC Subsecond	0-FFh	Unchanged	Read: only if RTCRE=1. Cannot be written. Cleared when RTCWE 1->0
RTCS.5-0	RTCS	RTC Second	0-3Bh	Unchanged	Read: only if RTCRE=1. Write: only if RTCWE=1. 1.95 ms Read/Write window
RTCM.5-0	RTCM	RTC Minute	0-3Bh	Unchanged	
RTCH.4-0	RTCH.4-0	RTC Hour	0-17h	Unchanged	
DOW.2-0	RTCH.7-5	RTC Day of Week	0-7h	Unchanged	Read: only if RTCRE=1. Write: only if RTCWE=1. 1.95 ms Read/Write window
RTCD1.7-0 RTCD0.7-0	RTCD1, (MSB) RTCD0, (LSB)	RTC Day	0-FFFFh	Unchanged	
SRCE	RTCC.7	RTC Subsecond Compare Enable		Unchanged	Unrestricted
SCE	RTCC.6	RTC Second Compare Enable		Unchanged	Unrestricted
MCE	RTCC.5	RTC Minute Compare Enable		Unchanged	Unrestricted
HCE	RTCC.4	RTC Hour Compare Enable		Unchanged	Unrestricted
RTCRES	RTCC.3	RTC Read Enable		0	Unrestricted
RTCWE	RTCC.2	RTC Write Enable		0	Read: Unrestricted Write: Timed Access
RTCIF	RTCC.1	RTC Interrupt Flag		Unchanged	Unrestricted
RTCE	RTCC.0	RTC Enable		Unchanged	Read: Unrestricted Write: Timed Access
E4K	TRIM.7	External 4096 Hz RTC Signal Enable		0	
X12/6	TRIM.6	RTC Crystal Capacitance Select		Unchanged	Read: Unrestricted Write: Timed Access
TRM2-0	TRIM.5 TRIM.3 TRIM.1	RTC Trim Bit 2-0		Unchanged	

RTCS;FBh, RTCM;FCh, RTCH;FDh, RTCD0;FEh, RTCD1;FFh), RTC alarm registers (RTASS;F2h, RTAS;F3h, RTAM;F4h, RTAH;F5h), RTC calibration (TRIM;96h), and RTC control (RTCC;F9h).

## STARTING AND STOPPING THE RTC

Operation of the RTC is enabled by setting the RTC Enable bit, RTCE (RTCC.0) to 1. This will start the RTC crystal amplifier, and begin clocking the RTC. Like all crystal oscillators, the RTC crystal oscillator has a crystal warm-up period. Software should allow a minimum of 1 second between setting the RTCE bit to 1 and initializing the time. This allows the clock to be guaranteed stable when timekeeping begins. Although it may be desired to program the RTC time registers and then start the oscillator, this sequence is not recommended because of the delay incurred by the RTC crystal warm-up period.

There are two situations where the RTC will be started. The first is the case where the RTC has been intentionally halted following normal operation. When the RTCE bit is set, the time registers will continue their count from the last setting when the clock was stopped. The RTC time value will be inaccurate, although the settings of the RTC alarm registers and the RTCC register will remain intact.

The second case is following the application of battery power. Most of the registers associated with the RTC are non-volatile, so that they will maintain their state while  $V_{CC}$  is removed. When battery power is applied to the device, however, the battery backed registers and bits associated with the RTC will be in an indeterminate state and will need to be reinitialized. This includes the RTC Interrupt Flag, RTCIF (RTCC.1), which should be cleared before setting the RTC Interrupt Enable bit (EIE.5).

The RTC can be halted by clearing the RTCE bit to 0. This will immediately halt the RTC and will freeze all the time registers at their current value and preserve all the RTC settings. If RTC functions are not desired, this can be used to reduce the power consumption of the device while in battery backed mode.

Access to the RTC time registers (RTCSS, RTCS, RTCM, RTCH, RTCD0, RTCD1) is enabled by the RTCRE (RTCC.3) and RTCWE (RTCC.2) bits. Both user software and the internal clock directly write and read the RTC time. To prevent the possibility of both user software and the internal timer accessing the same register simultaneously, the DS87C530 incorporates a register locking mechanism. Updates to the RTC time registers by the internal timer are temporarily suspended for up to 1.95 ms during software read or write operations. If a subsecond timer tick should occur during the 1.95 ms window, it will be processed immediately as soon as either the RTCWE or RTCRE bit is cleared. Because the subsecond timer tick interval is 3.906 ms, the 1.95 ms window allows sufficient time to complete any operations and process suspended timer ticks before the next timer tick occurs. In this way, no timer ticks can be lost, and accessing the time registers will not affect the accuracy of the RTC. To allow any pending timer ticks to propagate through the RTC circuitry, software must wait 4 machine cycles after setting the RTCWE or RTCRE bits before accessing any of the RTC time registers.

Reading the current time from any or all of the RTC time registers is accomplished by the following procedure:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Set the RTCRE bit (RTCC.3),
3. Wait 4 machine cycles,
4. Read the appropriate register(s) within 1 ms of RTCRE being set,
5. Clear the RTCRE bit (RTCC.3),
6. Enable interrupts by setting the EA bit (IE.7).

The time on the DS87C530 is set by writing to the Clock Registers. The Second, Minute, Hour, Day of the Week, and Day Count can be set by writing to the respective registers. It is not possible to set the Subsecond Real Time Clock Register (RTCS;FAh). This register is automatically reset to 00h when the RTCWE bit is cleared, either through software or the automatic time-out of the 1.95 ms write window. Writing an invalid time to these registers (loading the RTCM register with 3Dh or 61 minutes, for example) will result in an inaccurate count

by the RTC. It is the responsibility of the software to ensure that only valid times are written to these registers.

The procedure for setting an RTC time register is as follows:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Perform a Timed Access procedure,
3. Set the RTCWE bit (RTCC.2),
4. Wait 4 machine cycles,
5. Write the appropriate register(s) within 1.95 ms of RTCWE being set,
6. Perform a Timed Access procedure,
7. Clear the RTCWE bit (RTCC.2),
8. Enable interrupts by setting the EA bit (IE.7).

### USING THE RTC ALARM

The RTC alarm function is used to generate an interrupt when the RTC value matches selected alarm register values. An alarm can be triggered by a match on one or more of the following alarm registers: Subsecond (RTASS; F2h), Second (RTAS; F3h), Minute (RTAM; F4h), and Hour (RTAH; F5h). Note that there is no alarm register associated with the RTC Day Count or Day of Week registers. If an alarm is desired on a specific date, an alarm can be executed once a day and user software can compare the current date against the Day Register. It is not necessary to set the RTC Write Enable bit, RTCWE, when setting the alarm registers.

The alarm can be set to occur on a match with any or all of the alarm registers. An alarm can occur on a unique time of day, or a recurring alarm can be programmed every subsecond, second, minute, or hour. Alarms can occur synchronously, when the clock rolls over to match the alarm condition, or asynchronously, if the alarm registers are set to a value that matches the current time. Note that only one alarm may occur per subsecond tick. This means that if a synchronous alarm has already occurred during the current subsecond, software cannot cause an asynchronous alarm in the same subsecond.

The specific alarm registers to be compared are selected by setting or clearing the corresponding compare enable bits (RTCC.7-4). Any compare bit which is cleared will result in that register being treated as a Don't Care when evaluating alarm conditions. Clearing

all the compare enable bits will disable the ability of the RTC to cause an interrupt, and will immediately clear the RTC Interrupt Flag (RTCC.1). Unlike some interrupts, the RTC flag is not cleared by exiting the RTC interrupt service routine and must be explicitly cleared in software.

The general procedure for setting the RTC alarm registers to cause a RTC interrupt is as follows:

1. Clear the ERTCI Enable bit (EIE.5),
2. Clear all RTC Alarm Compare enable bits (ANL RTCC, #0Fh),
3. Write one or more RTC Alarm registers,
4. Set the desired RTC Alarm Compare enable bits,
5. Set the ERTCI Enable bit (EIE.5).

Setting the alarm to cause an interrupt once during a 24-hour period is done by setting all the alarm registers to the desired value and enabling all compare bits. A recurring alarm is enabled by clearing the compare enable bits associated with one or more alarm registers. For example, to specify an alarm to occur once a minute, the SSCE and SCE bits would be set. In general, a recurring alarm is set using the next lower time increment than the desired interrupt period. For example, if an alarm was desired once an hour, on the hour, a compare on the Real Time Alarm Minute Register would be performed, because the Real Time Clock Minute Register will match the corresponding alarm register only once an hour. The RTASS, RTAS, and RTAM registers would be cleared to 00h, and the SSCE, SCE, and MCE bits would all be set to 1 to match on the time xx:00:00. Writing an invalid time to these registers (loading the RTAM register with 3Dh or 61 minutes, for example) will never cause a match by the RTC. It is the responsibility of the software to ensure that only valid times are written to these registers.

It is important to remember that any RTC register whose corresponding compare enable bit is cleared to 0 will always be treated as a match. The alarm registers are interrogated once per subsecond tick to check for an alarm condition. If the SSCE bit was set to a don't care (cleared to 0) in the above example, a match (and interrupt) would occur during every subsecond of the minute in which the Real Time Alarm Minute register matched.

If an alarm occurs while in data retention state ( $V_{CC} < V_{BAT}$ ), the RTCIF flag will be set and the interrupt will remain pending. When power is reapplied to the device, the device will execute an RTC interrupt as soon as interrupts are enabled.

### USING THE DAY OF THE WEEK BITS

The DS87C530 contains 3 Day of the Week bits, DOW.2–0 located in the upper 3 bits of the Real Time Clock Hour register (RTCH;FDh). These allow the processor to count from 1 to 7. The day of the week bits will increment anytime the hour register changes from 17h to 00h, indicating a new day. When the day of the week register reaches a count of 111b, it will roll over to 001b.

If the day of the week feature is not needed, writing 000b to the bits will disable the ability of an hour register rollover to change the day of the week. The bits will remain at 000b. This is very convenient from a software standpoint, as it is not necessary to zero out the high order bits when determining the hour from the RTC Hour register.

### CHOOSING AN RTC CRYSTAL

The RTC clock source is provided by an external 32.768 kHz crystal attached to the RTCX1 and RTCX2 leads of the DS87C530. The device can be programmed to operate with a crystal rated for either a 6 pF or 12.5 pF load capacitance. The crystal selection is determined by the RTC Crystal Capacitance Select bit (TRIM.6). The default state of this bit after a no-battery reset is for a 12.5 pF crystal.

In general, a lower capacitance crystal will consume less power, but will be more susceptible to noise. Unlike the processor crystal inputs (X1, X2), the RTC crystal does not require external load capacitors. Placing load capacitors on the RTC crystal input pins will cause the RTC to keep incorrect time. To prevent system noise

from affecting the RTC, the RTCX1 and RTCX2 pins should be guard-ringed with the GND2 signal.

### CALIBRATING THE RTC OSCILLATOR

Although the DS87C530 RTC accuracy is guaranteed for  $\pm 2$  minutes per month, users may occasionally require greater accuracy. The RTC incorporates the ability to adjust the internal capacitance of the crystal amplifier via the RTC Trim Bits (TRM2–0 and TRM2–0). This allows the user to more accurately match the capacitance of the crystal amplifier to the crystal. Note that under most circumstances no adjustment of the RTC crystal capacitance is necessary, as it will default to a minimum accuracy of  $\pm 2$  minutes per month.

All of the crystal capacitance controls are located in the RTC Trim register (TRIM;96h). Setting the E4K bit will enable the output of a 4096 Hz signal on P1.7. This signal is derived from a divide by 8 of the 32.768 kHz crystal. Because this is directly generated from the RTC, it can be used to determine the actual frequency of the RTC. By adjusting the value of the TRMx bits, the internal capacitance of the RTC can be varied, slightly slowing or speeding up the RTC frequency. The combination of TRMx bits (TRIM.5–0) that causes the output on pin P1.7 to most closely approximate 4096 Hz will provide the most accurate setting of RTC capacitance.

As a precaution against accidental corruption of the oscillator trim bit settings, the TRMx bits must be programmed in the same instruction to the inverse of their respective TRMx bits. For example, if a trim bit setting of 5 (101) was desired, the TRMx bits should be set to 2 (010). An illegal combination will automatically reset the TRIM register to 0x100101b. This will disable the E4K signal on P1.7, but leave the X12/6 bit unmodified.

More information on calibrating the RTC oscillator for improved accuracy is located in Application Note 79, Using the DS87C530 Real Time Clock.



## SECTION 15: BATTERY BACKUP

The DS87C530 incorporates a feature which can maintain timekeeping and on-chip SRAM contents in the absence of  $V_{CC}$ . An external energy source such as a lithium battery or 0.47 F super cap can be connected to the  $V_{BAT}$  pin. The nominal battery voltage should be 3V. For proper operation, the battery voltage must always be at least a diode drop (0.7V) below  $V_{CC}$ , and is recommended to be below  $V_{RST}$ .

The DS87C530 will automatically enter data retention mode when  $V_{CC} < V_{BAT}$ . When in data retention mode, the RTC and SRAM contents are powered from the energy source connected to the  $V_{BAT}$  pin and electrically isolated from the rest of the device. This means that writes to battery backed SFRs and SRAM are ignored and reads will return erroneous data while in data retention mode. The DS87C530 Data Sheet contains a functional diagram of the internal battery switching circuitry.

The data retention switch voltage, the point at which the device switches into data retention mode, is a function of the battery voltage, not an absolute reference. Care must be taken when selecting a battery so that its voltage will stay below  $V_{CC}$  during normal operation to prevent an unplanned lockout of the RTC and SRAM. Although it is unlikely that such a situation would occur, it could become an issue if a relatively high voltage battery is used. For example, suppose a 4.5 V battery is used with a device operating at a  $V_{CC}$  of 5.0 V. During normal operation,  $V_{CC}$  will be above  $V_{BAT}$ , so no prob-

lem will occur. Suppose that a loss of power occurs, and  $V_{CC}$  begins to drop. Under normal circumstances, the device will continue to operate until it reaches  $V_{RST}$  (4.0V to 4.25V), at which time device operation will halt. If  $V_{BAT}$  is higher than  $V_{RST}$ , however, RTC and SRAM access will be prohibited before the device enters reset. This means that there may be a short period of time before reset when the device is operating but could read erroneous data from the RTC or SRAM or fail to write to them. One solution would be to use the power-fail interrupt to halt reads or writes to the RTC or SRAM when  $V_{CC}$  is dropping. The best approach is to carefully select battery voltages to avoid the problem entirely.

## SELECTING A BATTERY

There are a number of battery chemistries and brands that are suitable for use with battery-backed members of the High-Speed Microcontroller Family. The use of lithium chemistry batteries, such as Lithium Manganese Dioxide, is preferred as their nominal voltage is approximately 3.0V. Coin cells are particularly suited for use with the High-Speed Microcontroller Family because of their capacity, low profile, and small diameter. Many are available with PC mount tabs attached for automated assembly. Table 15-1 shows a list of some common batteries and their capacities. This list is by no means exhaustive, and the inclusion or exclusion of any vendor from this list is in no way a comment on the suitability of a specific battery in a customer's application.

**SUGGESTED BATTERIES FOR USE WITH DS87C530** Table 15-1

MANUFACTURER	MODEL NUMBER	TYPE	NOMINAL VOLTAGE	CAPACITY
Panasonic PH: (201) 348-5266	CR1620	Lithium/Manganese Dioxide	3 V	70 mAh
	CR1616	Lithium/Manganese Dioxide	3 V	50 mAh
	CR1220	Lithium/Manganese Dioxide	3 V	35 mAh
	BR1616	Lithium/poly-carbon monofluoride	3 V	48 mAh
	BR1225	Lithium/poly-carbon monofluoride	3 V	38 mAh



Battery life can be calculated by dividing the rated battery capacity by the  $I_{BAT}$  current specified on the device specific data sheet. Note that this determines the minimum battery life; while  $V_{CC}$  is applied to the device, it draws negligible current from the battery, and so battery life will be lengthened accordingly. Backup current is a function of temperature, and therefore battery life is dependent on the operating environment.

The registers shown in Table 15–2 are battery-backed, and one or more bits will be indeterminate following a no-battery reset. They should be initialized as part of a no-battery reset procedure.

**BATTERY BACKED SFRS** Table 15–2

REGISTER NAME	LOCATION
TRIM	96h
RTASS	F2h
RTAS	F3h
RTAM	F4h
RTAH	F5h
RTCC	F9h
RTCSS	FAh
RTCS	FBh
RTCM	FCh
RTCH	FDh
RTCD0	FEh
RTCD1	FFh

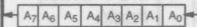
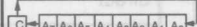
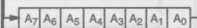
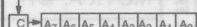
## LITHIUM BATTERY CONSIDERATIONS

Lithium primary batteries (non-rechargeable) can fail and/or rupture if subjected to reverse current from the device they are powering. The battery-switching circuitry inside the DS87C530 was designed to reduce or eliminate the need for external hardware required to meet battery safety regulations. As shown in the DS87C530 Data Sheet, a current limiting resistor is always in series with a switching field-effect transistor, regardless of whether the DS87C530 is drawing current from  $V_{CC}$  or  $V_{BAT}$  pins. This satisfies the two mechanism requirement of most safety codes.

## SECTION 16: INSTRUCTION SET DETAILS

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
ARITHMETIC OPERATION	ADD A, Rn	0	0	1	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	28-2F	1	1	(A) = (A) + (Rn)
	ADD A, direct	0	0	1	0	0	1	0	1	25	2	2	(A) = (A) + (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	ADD A, @Ri	0	0	1	0	0	1	1	i	26-27	1	1	(A) = (A) + ((Ri))
	ADD A, #data	0	0	1	0	0	1	0	0	24	2	2	(A) = (A) + #data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	ADDC A, Rn	0	0	1	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	38-3F	1	1	(A) = (A)+(C)+(Rn)
	ADDC A, direct	0	0	1	1	0	1	0	1	35	2	2	(A) = (A)+(C)+(direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	ADDC A, @Ri	0	0	1	1	0	1	1	i	36-37	1	1	(A) = (A)+(C)+((Ri))
	ADDC A, #data	0	0	1	1	0	1	0	0	34	2	2	(A) = (A)+(C)+#data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	SUBB A, Rn	1	0	0	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	98-9F	1	1	(A) = (A)-(C)-(Rn)
	SUBB A, direct	1	0	0	1	0	1	0	1	95	2	2	(A) = (A)-(C)-(direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	SUBB A, @Ri	1	0	0	1	0	1	1	i	96-97	1	1	(A) = (A)-(C)-((Ri))
	SUBB A, #data	1	0	0	1	0	1	0	0	94	2	2	(A) = (A)-(C)-#data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	INC A	0	0	0	0	0	1	0	0	04	1	1	(A) = (A) + 1
	INC Rn	0	0	0	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	08-0F	1	1	(Rn) = (Rn) + 1
	INC direct	0	0	0	0	0	1	0	1	05	2	2	(direct) = (direct)+1
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	INC @Ri	0	0	0	0	0	1	1	i	06-07	1	1	((Ri)) = ((Ri)) + 1
	INC DPTR	1	0	1	0	0	0	1	1	A3	1	3	(DPTR)=(DPTR)+1
	DEC A	0	0	0	1	0	1	0	0	14	1	1	(A) = (A) - 1
	DEC Rn	0	0	0	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	18-1F	1	1	(Rn) = (Rn) - 1
	DEC direct	0	0	0	1	0	1	0	1	15	2	2	(direct) = (direct)-1
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	DEC @Ri	0	0	0	1	0	1	1	i	16-17	1	1	((Ri)) = ((Ri)) - 1
	MUL AB	1	0	1	0	0	1	0	0	A4	1	5	(B <sub>15-8</sub> ), (A <sub>7-0</sub> ) = (A) X (B)
	DIV AB	1	0	0	0	0	1	0	0	84	1	5	(A <sub>15-8</sub> ), (A <sub>7-0</sub> ) = (A) ÷ (B)

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
ARITHMETIC OPER.	DA A	1	1	0	1	0	1	0	0	D4	1	1	Contents of Accumulator are BCD. IF $[(A_{3-0}) > 9]$ OR $[(C) = 1]$ THEN $(A_{3-0}) = (A_{3-0}) + 6$ AND IF $[(A_{7-4}) > 9]$ OR $[(C) = 1]$ THEN $(A_{7-4}) = (A_{7-4}) + 6$
	ANL A, Rn	0	1	0	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	58-5F	1	1	$(A) = (A) \text{ AND } (Rn)$
	ANL A, direct	0	1	0	1	0	1	0	1	55	2	2	$(A) = (A) \text{ AND } (\text{direct})$
	ANL A, @Ri	0	1	0	1	0	1	1	1	56-57	1	1	$(A) = (A) \text{ AND } ((Ri))$
	ANL A, #data	0	1	0	1	0	1	0	0	54	2	2	$(A) = (A) \text{ AND } \#data$
	ANL direct, A	0	1	0	1	0	0	1	0	52	2	2	$(\text{direct}) = (\text{direct}) \text{ AND } A$
	ANL direct, #data	0	1	0	1	0	0	1	1	53	3	3	$(\text{direct}) = (\text{direct}) \text{ AND } \#data$
	ORL A, Rn	0	1	0	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	48-4F	1	1	$(A) = (A) \text{ OR } (Rn)$
	ORL A, direct	0	1	0	0	0	1	0	1	45	2	2	$(A) = (A) \text{ OR } (\text{direct})$
	ORL A, @Ri	0	1	0	0	0	1	1	i	46-47	1	1	$(A) = (A) \text{ OR } ((Ri))$
LOGICAL OPERATION	ORL A, #data	0	1	0	0	0	1	0	0	44	2	2	$(A) = (A) \text{ OR } \#data$
	ORL direct, A	0	1	0	0	0	0	1	0	42	2	2	$(\text{direct}) = (\text{direct}) \text{ OR } (A)$
	ORL direct, #data	0	1	0	0	0	0	1	1	43	3	3	$(\text{direct}) = (\text{direct}) \text{ OR } \#data$
	XRL A, Rn	0	1	1	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	68-6F	1	1	$(A) = (A) \text{ XOR } (Rn)$
	XRL A, direct	0	1	1	0	0	1	0	1	65	2	2	$(A) = (A) \text{ XOR } (\text{direct})$
	XRL A, @Ri	0	1	1	0	0	1	1	i	66-67	1	1	$(A) = (A) \text{ XOR } ((Ri))$
	XRL A, #data	0	1	1	0	0	1	0	0	64	2	2	$(\text{direct}) = (A) \text{ XOR } \#data$
	XRL direct, A	0	1	1	0	0	0	1	0	62	2	2	$(\text{direct}) = (\text{direct}) \text{ XOR } (A)$
	XRL direct, #data	0	1	1	0	0	0	1	1	63	3	3	$(\text{direct}) = (\text{direct}) \text{ XOR } \#data$
	CLR A	1	1	1	0	0	1	0	0	E4	1	1	$(A) = 0$
	CPL A	1	1	1	1	0	1	0	0	F4	1	1	$(A) = (\bar{A})$

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
LOGICAL OPERATION	RL A	0	0	1	0	0	0	1	1	23	1	1	 <p>The contents of the accumulator are rotated left by one bit.</p>
	RLC A	0	0	1	1	0	0	1	1	33	1	1	 <p>The contents of the accumulator are rotated left by one bit.</p>
	RR A	0	0	0	0	0	0	1	1	03	1	1	 <p>The contents of the accumulator are rotated right by one bit.</p>
	RRC A	0	0	0	1	0	0	1	1	13	1	1	 <p>The contents of the accumulator are rotated right by one bit.</p>
	SWAP A	1	1	0	0	0	1	0	0	C4	1	1	(A <sub>3-0</sub> ) $\leftrightarrow$ (A <sub>7-4</sub> )
	MOV A, Rn	1	1	1	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	E8-EF	1	1	(A) = (Rn)
DATA TRANSFER	MOV A, direct	1	1	1	0	0	1	0	1	E5	2	2	(A) = (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	MOV A, @Ri	1	1	1	0	0	1	1	i	E6-E7	1	1	(A) = ((Ri))
	MOV A, #data	0	1	1	1	0	1	0	0	74	2	2	(A) = #data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	MOV Rn, A	1	1	1	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	F8-FF	1	1	(Rn) = (A)
	MOV Rn, direct	1	0	1	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	A8-AF	2	2	(Rn) = (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	MOV Rn, #data	0	1	1	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	78-7F	2	2	(Rn) = #data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	MOV direct, A	1	1	1	1	0	1	0	1	F5	2	2	(direct) = (A)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	MOV direct, Rn	1	0	0	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	88-8F	2	2	(direct) = (Rn)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	MOV direct1, direct2	1	0	0	0	0	1	0	1	85	3	3	(direct1) = (direct2) (source) (destination)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 3			
	MOV direct, @Ri	1	0	0	0	0	1	1	i	86-87	2	2	(direct) = ((Ri))
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
DATA TRANSFER	MOV direct, #data	0	1	1	1	0	1	0	1	75	3	3	(direct) = #data
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 3			
	MOV @Ri, A	1	1	1	1	0	1	1	i	F6-F7	1	1	((Ri)) = A
	MOV @Ri, direct	1	0	1	0	0	1	1	i	A6-A7	2	2	((Ri)) = (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	MOV @Ri, #data	0	1	1	1	0	1	1	i	76-77	2	2	((Ri)) = #data
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
	MOV DPTR, #data16	1	0	0	1	0	0	0	0	90	3	3	(DPTR) = #data <sub>15-0</sub> (DPH) = #data <sub>15-8</sub> (DPL) = #data <sub>7-0</sub>
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 3			
	MOVC A, @A + DPTR	1	0	0	1	0	0	1	1	93	1	3	(A) = ((A) + (DPTR))
	MOVC A, @A + PC	1	0	0	0	0	0	1	1	83	1	3	(A) = ((A) + (PC))
	MOVX A, @Ri	1	1	1	0	0	0	1	i	E2-E3	1	2-9	(A) = ((Ri))
	MOVX @DPTR	1	1	1	0	0	0	0	0	E0	1	2-9	(A) = ((DPTR))
	MOVX @Ri, A	1	1	1	1	0	0	1	i	F2-F3	1	2-9	((Ri)) = (A)
	MOVX @DPTR, A	1	1	1	1	0	0	0	0	F0	1	2-9	((DPTR)) = (A)
DATA TRANSFER	PUSH direct	1	1	0	0	0	0	0	0	C0	2	2	(SP) = (SP) + 1 ((SP)) = (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	POP direct	1	1	0	1	0	0	0	0	D0	2	2	(direct) = ((SP)) (SP) = (SP) - 1
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	XCH A, Rn	1	1	0	0	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	C8-CF	1	1	(A) = (Rn)
	XCH A, direct	1	1	0	0	0	1	0	1	C5	2	2	(A) = (direct)
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	XCH A, @Ri	1	1	0	0	0	1	1	i	C6-C7	1	1	(A) = ((Ri))
	XCHD A, @Ri	1	1	0	1	0	1	1	i	D6-D7	1	1	(A <sub>3-0</sub> ) = ((Ri <sub>3-0</sub> ))



	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
BOOLEAN VARIABLE MANIPULATION	CLR C	1	1	0	0	0	0	1	1	C3	1	1	(C) = 0
	CLR bit	1	1	0	0	0	0	1	0	C2	2	2	(bit) = 0
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	SETB C	1	1	0	1	0	0	1	1	D3	1	1	(C) = 1
	SETB bit	1	1	0	1	0	0	1	0	D2	2	2	(bit) = 1
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	CPL C	1	0	1	1	0	0	1	1	B3	1	1	(C) = ( $\bar{C}$ )
	CPL bit	1	0	1	1	0	0	1	0	B2	2	2	(bit) = ( $\bar{\text{bit}}$ )
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	ANL C, bit	1	0	0	0	0	0	1	0	82	2	2	(C) = (C) AND (bit)
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	ANL C, $\bar{\text{bit}}$	1	0	1	1	0	0	0	0	B0	2	2	(C) = (C) AND ( $\bar{\text{bit}}$ )
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	ORL C, bit	0	1	1	1	0	0	1	0	72	2	2	(C) = (C) OR (bit)
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	ORL C, $\bar{\text{bit}}$	1	0	1	0	0	0	0	0	A0	2	2	(C) = (C) OR ( $\bar{\text{bit}}$ )
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	MOV C, bit	1	0	1	0	0	0	1	0	A2	2	2	(C) = (bit)
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
	MOV bit, C	1	0	0	1	0	0	1	0	92	2	2	(bit) = (C)
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
PROGRAM BRANCHING	ACALL addr 11	a <sub>10</sub>	a <sub>9</sub>	a <sub>8</sub>	1	0	0	0	1	Byte 1	2	3	(PC) = (PC) + 2 (SP) = (SP) + 1 ((SP)) = (PC <sub>7-0</sub> ) (SP) = (SP) + 1 ((SP)) = (PC <sub>15-8</sub> ) (PC) = page address
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	LCALL addr 16	0	0	0	1	0	0	1	0	12	3	4	(PC) = (PC) + 3 (SP) = (SP) + 1 ((SP)) = (PC <sub>7-0</sub> ) (SP) = (SP) + 1 ((SP)) = (PC <sub>15-8</sub> ) (PC) = addr <sub>15-0</sub>
		a <sub>15</sub>	a <sub>14</sub>	a <sub>13</sub>	a <sub>12</sub>	a <sub>11</sub>	a <sub>10</sub>	a <sub>9</sub>	a <sub>8</sub>	Byte 2			
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 3			
	RET	0	0	1	0	0	0	1	0	22	1	4	(PC <sub>15-8</sub> ) = ((SP)) (SP) = (SP) - 1 (PC <sub>7-0</sub> ) = ((SP)) (SP) = (SP) - 1
	RETI	0	0	1	1	0	0	1	0	32	1	4	(PC <sub>15-8</sub> ) = ((SP)) (SP) = (SP) - 1 (PC <sub>7-0</sub> ) = ((SP)) (SP) = (SP) - 1
	AJMP addr 11	a <sub>10</sub>	a <sub>9</sub>	a <sub>8</sub>	0	0	0	0	1	Byte 1	2	3	(PC) = (PC) + 2 (PC <sub>10-0</sub> ) = page addr
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
	LJMP addr 16	0	0	0	0	0	0	1	0	02	3	4	(PC) = addr <sub>15-0</sub>
		a <sub>15</sub>	a <sub>14</sub>	a <sub>13</sub>	a <sub>12</sub>	a <sub>11</sub>	a <sub>10</sub>	a <sub>9</sub>	a <sub>8</sub>	Byte 2			
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 3			
	SJMP rel	1	0	0	0	0	0	0	0	80	2	3	(PC) = (PC) + 2 (PC) = (PC) + rel
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 2			
	JMP @A + DPTR	0	1	1	1	0	0	1	1	73	1	3	(PC) = (A) + (DPTR)
	JZ rel	0	1	1	0	0	0	0	0	60	2	3	(PC) = (PC) + 2 IF (A) = 0 THEN (PC) = (PC) + rel
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 2			
	JNZ rel	0	1	1	1	0	0	0	0	70	2	3	(PC) = (PC) + 2 IF (A) ≠ 0 THEN (PC) = (PC) + rel
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 2			

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
PROGRAM BRANCHING	JC rel	0	1	0	0	0	0	0	0	40	2	3	(PC) = (PC) + 2 IF (C) = 1 THEN (PC) = (PC) + rel
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 2			
	JNC rel	0	1	0	1	0	0	0	0	50	2	3	(PC) = (PC) + 2 IF (C) ≠ 0 THEN (PC) = (PC) + rel
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 2			
	JB bit, rel	0	0	1	0	0	0	0	0	20	3	4	(PC) = (PC) + 3 IF (bit) = 1 THEN (PC) = (PC) + rel
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	JNB bit, rel	0	0	1	1	0	0	0	0	30	3	4	(PC) = (PC) + 3 IF (bit) = 0 THEN (PC) = (PC) + rel
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	JBC bit, direct rel	0	0	0	1	0	0	0	0	10	3	4	(PC) = (PC) + 3 IF (bit) = 1 THEN (bit) = 0 (PC) = (PC) + rel
		b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	CJNE A, direct rel	1	0	1	1	0	1	0	1	B5	3	4	(PC) = (PC) + 3 IF (direct) < (A) THEN (PC) = (PC) + rel and (C) = 0 OR IF (direct) > (A) THEN (PC) = (PC) + rel and (C) = 1
		a <sub>7</sub>	a <sub>6</sub>	a <sub>5</sub>	a <sub>4</sub>	a <sub>3</sub>	a <sub>2</sub>	a <sub>1</sub>	a <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	CJNE A, #data rel	1	0	1	1	0	1	0	0	B4	3	4	(PC) = (PC) + 3 IF #data < (A) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > (A) THEN (PC) = (PC) + rel and (C) = 1
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	CJNE Rn, #data rel	1	0	1	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	B8-BF	3	4	(PC) = (PC) + 3 IF #data < (Rn) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > (Rn) THEN (PC) = (PC) + rel and (C) = 1
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			
	CJNE @Ri, #data rel	1	0	1	1	0	1	1	i	B6-B7	3	4	(PC) = (PC) + 3 IF #data < ((Ri)) THEN (PC) = (PC) + rel and (C) = 0 OR IF #data > ((Ri)) THEN (PC) = (PC) + rel and (C) = 1
		d <sub>7</sub>	d <sub>6</sub>	d <sub>5</sub>	d <sub>4</sub>	d <sub>3</sub>	d <sub>2</sub>	d <sub>1</sub>	d <sub>0</sub>	Byte 2			
		r <sub>7</sub>	r <sub>6</sub>	r <sub>5</sub>	r <sub>4</sub>	r <sub>3</sub>	r <sub>2</sub>	r <sub>1</sub>	r <sub>0</sub>	Byte 3			

	MNEMONIC	INSTRUCTION CODE								HEX	BYTE	CYCLE	EXPLANATION
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				
PROGRAM BRANCHING	DJNZ Rn, rel	1	1	0	1	1	n <sub>2</sub>	n <sub>1</sub>	n <sub>0</sub>	D8-Df Byte 2	2	3	(PC) = (PC) + 2 (Rn) = (Rn) - 1 IF (Rn) 0 THEN (PC) = (PC) + rel
	DJNZ direct rel	1	1	0	1	0	1	0	1	D5 Byte 2 Byte 3	3	4	(PC) = (PC) + 3 (direct) = (direct) - 1 IF (direct) ≠ 0 THEN (PC) = (PC) + rel
	NOP	0	0	0	0	0	0	0	0	00	1	1	(PC) = (PC) + 1

PROGRAM BRANCHING	JNB rel	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel
	JNC rel	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel
	JNB direct	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel
	JNC direct	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel
	JNB A, direct	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel
	JNC A, direct	0	0	1	0	0	0	0	0	00	1	1	(PC) = (PC) + 1 IF (Rn) = 0 THEN (PC) = (PC) + rel

## SECTION 17: TROUBLESHOOTING

### DEVICE OPERATES AT 1/3 OF CRYSTAL SPEED

The High Speed Microcontroller Family operates from the primary or fundamental mode of the external crystal. Many off-the-shelf high-frequency crystals are specified to operate from their third overtone. When used with a High-Speed Microcontroller, these crystals will resonate in their primary mode, which will appear to be 1/3 of the rated crystal speed. Make sure that any crystals used will operate at their rated speed in primary mode.

### DEVICE RESETS FOR NO REASON

During the debugging process, it may be necessary to isolate the cause of an unexpected device reset. Because resets are initiated by a limited number of sources, it is relatively easy to determine their source by interrogating a few bits. These bits should be interrogated early in the code following a reset to determine its source. As a debug tool, software could set the state of one or more port pins to indicate the type of reset to the designer. Note that power supply problems or glitches will appear as unplanned power-on resets.

SOURCE	POR BIT WDCON.6	WTRF BIT WDCON.3
Power-on reset	1	0
Watchdog reset	0	1
External reset	0	0

### ACCESS TO INTERNAL MOVX SRAM IS UNSUCCESSFUL

The internal MOVX SRAM available on some members of the High Speed Microcontroller Family is disabled after any reset. To enable the on-chip SRAM, the software should configure the Data Memory Enable bits (PMR.1–0) as needed.

When  $V_{CC}$  drops below  $V_{BAT}$ , access to the SRAM is disabled to prevent corruption of the data. If the battery voltage is greater than  $V_{RST}$ , this means that the processor may continue to operate while SRAM access is denied. Make sure that the battery voltage remains below the minimum  $V_{RST}$ .

### REAL-TIME CLOCK DOES NOT OPERATE OR KEEP ACCURATE TIME.

The state of the real-time clock (RTC) used on the DS87C530 is undefined following a no-battery reset or battery attach. The RTC oscillator must be enabled by setting the RTCE bit (RTCC.0) for the RTC to work.

The RTC is guaranteed to a minimum accuracy of  $\pm 2$  minutes per month over the rated temperature and voltage specifications. If the time is found to be less accurate than this, it is most likely due to the selection of crystal. Make sure that the RTC crystal is 32.768 kHz, and either 12.5 pF or 6 pF capacitance. The 12/6 bit (TRIM.6) setting should correspond to the crystal in use. Unlike other crystals, external load capacitors should not be used with the RTC. These will seriously distort the accuracy of the clock. Additional information on design considerations with the RTC can be found in Applications Note 79, Using the DS87C530 Real Time Clock.

### SERIAL PORT DOES NOT WORK

The serial port is not a complicated peripheral, but there are many elements that need to be initialized. The following checklist is provided to help in debugging.

1. Have the appropriate port latch bits (P3.0, P3.1, P1.2, or P1.3) been set to 1 to enable the serial port functions?
2. Has the correct timebase been selected? (4 clocks per tick or 12 clocks per tick)
3. Is the appropriate timer reload value loaded?
4. Is the appropriate timer mode selected?
5. Is the appropriate timer running by setting TR0, TR1, or TR2 bits?
6. Is the correct serial port mode selected?
7. If desired, is the serial port doubler bit, SMOD, set? (PCON.7 or WDCON.7)
8. If desired, is the receive enable bit (REN\_0 or REN\_1) set?
9. Is the serial port interrupt enabled?
10. Is the global interrupt enable bit set?



## HIGH-SPEED MICROCONTROLLER DOES NOT WORK IN EXISTING 8051 DESIGN

Although the High Speed Microcontroller Family was designed as a drop in replacement for the 8051 family, occasionally a developer may notice problems when inserting into an existing design. Often these problems are related to slow memory interfaces which cannot keep up with the increased throughput of the faster microcontroller. In addition, software timing loops will run

faster, possibly changing program operation. These and other effects are described in Application Note 56 (The DS80C320 as a Drop-In Replacement for the 8032). Application Note 57 (DS80C320 Memory Interface Timing) discusses memory interface timing for the DS80C320. Application Note 89 (High-Speed Micro Memory Interface Timing) discusses interfacing other members of the High-Speed Microcontroller family to external memory.

## DEVICE RESETS FOR NO REASON

During the debugging process, it may be necessary to isolate the cause of an unexpected device reset. Because resets are initiated by a limited number of sources, it is relatively easy to determine their source by interrogating a few bits. These bits should be interrogated early in the code following a reset to determine the source. As a debug tool, software could set the state of one or more port pins to indicate the type of reset to the designer. Note that power supply problems or glitches will appear as unplanned power-on resets.

SOURCE	POR BIT WDON.6	WTRF BIT WDON.3
Power-on reset	1	0
Watchdog reset	0	1
External reset	0	0

## ACCESS TO INTERNAL MOVX SRAM IS UNSUCCESSFUL

The internal MOVX SRAM available on some members of the High-Speed Microcontroller Family is disabled after any reset. To enable the on-chip SRAM, the software should configure the Data Memory Enable bit (PMR.1-0) as needed.

When V<sub>CC</sub> drops below V<sub>BAT</sub>, access to the SRAM is disabled to prevent corruption of the data. If the battery voltage is greater than V<sub>BAT</sub>, this means that the processor may continue to operate while SRAM access is denied. Make sure that the battery voltage remains below the minimum V<sub>rest</sub>.

## SERIAL PORT DOES NOT WORK

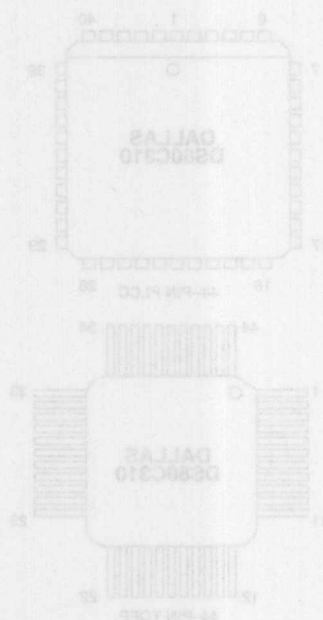
The serial port is not a complicated peripheral, but there are many elements that need to be initialized. The following checklist is provided to help in debugging.

1. Have the appropriate port latch bits (P3.0, P3.1, P1.2, or P1.3) been set to 1 to enable the serial port functions?
2. Has the correct baudrate been selected? (4 clock port tick or 12 clock port tick)
3. Is the appropriate timer reload value loaded?
4. Is the appropriate timer mode selected?
5. Is the appropriate timer running by setting TR0, TR1, or TR2 bits?
6. Is the correct serial port mode selected?
7. If desired, is the serial port double bit (SMOD, SMOD.7) (PCON.7 or WDON.7)?
8. If desired, is the receive enable bit (REN, 0 or REN.1) set?
9. Is the serial port interrupt enabled?
10. Is the global interrupt enable bit set?

# DS80C310 High-Speed Micro

## DATA SHEETS SEMICONDUCTOR

### PACKAGE OUTLINE



### FEATURES

- 80C32 Compatible
  - 8081 pin and instruction set compatible
  - Full duplex serial port
  - Three 16-bit timers/counters
  - 32K bytes scratchpad RAM
  - Multiplexed address/data bus
  - Addressed 8-KB ROM and 8-KB RAM
- High-Speed Architecture
  - 1 clock-machine cycle (8081 = 12)
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Dual data pointer
  - Optional variable length MOVX to access fast slow RAM peripherals
- 10 total interrupt sources with 8 external
- Internal power on reset circuit
- Fully compatible with the DS80C320
- Available in 44-pin PDIP, 44-pin PLCC, and 44-pin TOPP

### DESCRIPTION

The DS80C310 is a fast 80C32-compatible microcontroller. It features a redesigned processor without wasted clock and memory cycles. As a result, it executes every 8081 instruction between 1.8 and 3 times faster than the original architecture for the same crystal speed. Typical applications will see a speed improvement of 5 times using the same code and the same crystal. The DS80C310 offers a maximum crystal speed of 33 MHz, resulting in equivalent execution speeds of 85.5 MHz (approximately 2.5X).

# DALLAS

SEMICONDUCTOR

## DS80C310

### High-Speed Micro

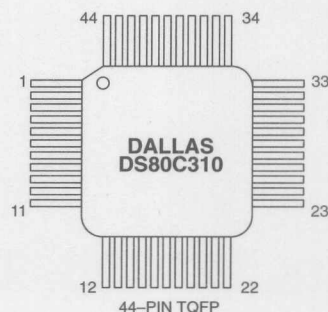
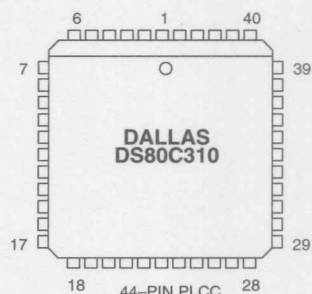
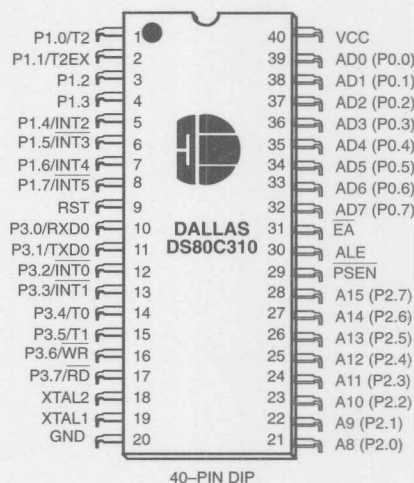
#### FEATURES

- 80C32 Compatible
  - 8051 pin and instruction set compatible
  - Full duplex serial port
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
  - Multiplexed address/data bus
  - Addresses 64KB ROM and 64KB RAM
- High-Speed Architecture
  - 4 clocks/machine cycle (8051 = 12)
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM /peripherals
- 10 total interrupt sources with 6 external
- Internal power on reset circuit
- Upwardly compatible with the DS80C320
- Available in 40-pin PDIP, 44-pin PLCC, and 44-pin TQFP

#### DESCRIPTION

The DS80C310 is a fast 80C31/80C32 compatible microcontroller. It features a redesigned processor core without wasted clock and memory cycles. As a result, it executes every 8051 instruction between 1.5 and 3 times faster than the original architecture for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and the same crystal. The DS80C310 offers a maximum crystal speed of 33 MHz, resulting in apparent execution speeds of 82.5 MHz (approximately 2.5X).

#### PACKAGE OUTLINE



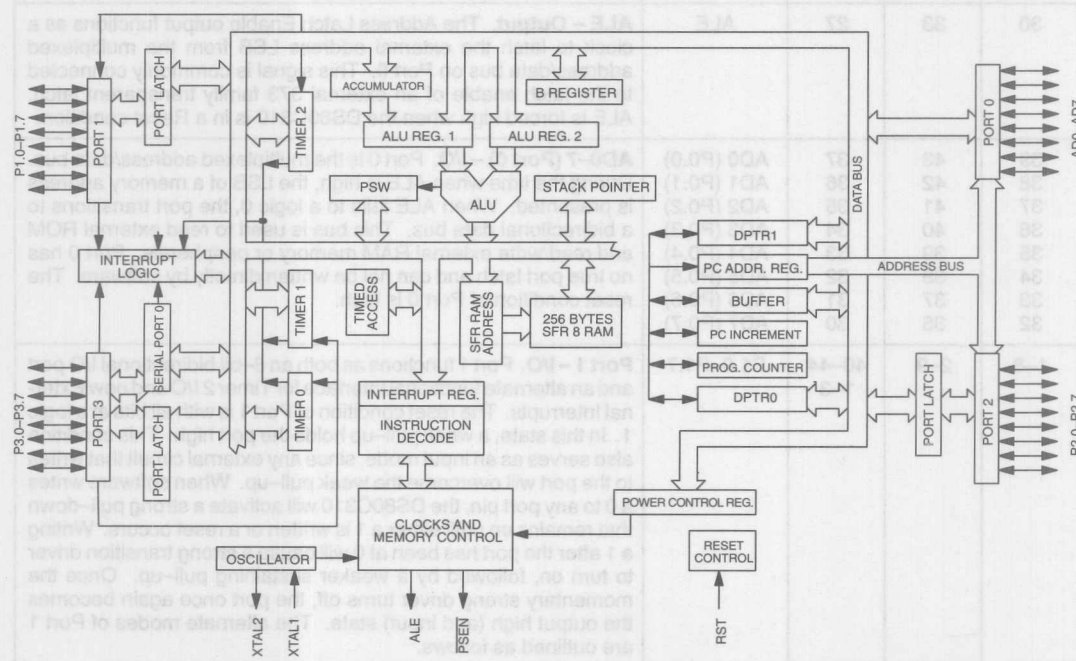
The DS80C310 is pin compatible with the standard 80C32 and includes standard resources such as three timer/counters, 256 bytes of RAM, and a serial port. It also provides dual data pointers (DPTRs) to speed block data memory moves. It also can adjust the speed

of MOVX data memory access between two and nine machine cycles for flexibility in selecting external memory and peripherals. The DS80C310 offers upward compatibility with the DS80C320.

#### ORDERING INFORMATION:

PART NUMBER	PACKAGE	MAX. CLOCK SPEED	TEMPERATURE RANGE
DS80C310-MCG	40-pin plastic DIP	25 MHz	0°C to 70°C
DS80C310-QCG	44-pin PLCC	25 MHz	0°C to 70°C
DS80C310-ECG	44-pin TQFP	25 MHz	0°C to 70°C
DS80C310-MCL	40-pin plastic DIP	33 MHz	0°C to 70°C
DS80C310-QCL	44-pin PLCC	33 MHz	0°C to 70°C
DS80C310-ECL	44-pin TQFP	33 MHz	0°C to 70°C

DS80C310 BLOCK DIAGRAM Figure 1



PIN DESCRIPTION Table 1

DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION
40	44	38	V <sub>CC</sub>	V <sub>CC</sub> – +5V.
20	22, 23, 1	16, 17, 39	GND	GND – Digital circuit ground.
9	10	4	RST	<b>RST – Input.</b> The RST input pin contains a Schmitt voltage input to recognize external active high reset inputs. The pin also employs an internal pull-down resistor to allow for a combination of wired OR external Reset sources.
18 19	20 21	14 15	XTAL2 XTAL1	<b>XTAL1, XTAL2</b> – The crystal oscillator pins XTAL1 and XTAL2 provide support for parallel resonant, AT cut crystals. XTAL1 acts also as an input in the event that an external clock source is used in place of a crystal. XTAL2 serves as the output of the crystal amplifier.
29	32	26	PSEN	<b>PSEN – Output.</b> The Program Store Enable output. This signal is commonly connected to external ROM memory as a chip enable. PSEN is active low. PSEN is driven high when data memory (RAM) is being accessed through the bus and during a reset condition.
30	33	27	ALE	<b>ALE – Output.</b> The Address Latch Enable output functions as a clock to latch the external address LSB from the multiplexed address/data bus on Port 0. This signal is commonly connected to the latch enable of an external 373 family transparent latch. ALE is forced high when the DS80C310 is in a Reset condition.
39 38 37 36 35 34 33 32	43 42 41 40 39 38 37 36	37 36 35 34 33 32 31 30	AD0 (P0.0) AD1 (P0.1) AD2 (P0.2) AD3 (P0.3) AD4 (P0.4) AD5 (P0.5) AD6 (P0.6) AD7 (P0.7)	<b>AD0–7 (Port 0) – I/O.</b> Port 0 is the multiplexed address/data bus. During the time when ALE is high, the LSB of a memory address is presented. When ALE falls to a logic 0, the port transitions to a bidirectional data bus. This bus is used to read external ROM and read/write external RAM memory or peripherals. Port 0 has no true port latch and can not be written directly by software. The reset condition of Port 0 is high.
1–8	2–9	40–44 1–3	P1.0–P1.7	<b>Port 1 – I/O.</b> Port 1 functions as both an 8-bit bidirectional I/O port and an alternate functional interface for Timer 2 I/O and new External Interrupts. The reset condition of Port 1 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS80C310 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port once again becomes the output high (and input) state. The alternate modes of Port 1 are outlined as follows.



DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION	
				Port	Alternate Function
1	2	40		P1.0	T2 External I/O for Timer/Counter 2
2	3	41		P1.1	T2EX Timer/Counter 2 Capture/Reload Trigger
3	4	42		P1.2	none (DS80C320 has a serial port RXD)
4	5	43		P1.3	none (DS80C320 has a serial port TXD)
5	6	44		P1.4	INT2 External Interrupt 2 (Positive Edge Detect)
6	7	1		P1.5	INT3 External Interrupt 3 (Negative Edge Detect)
7	8	2		P1.6	INT4 External Interrupt 4 (Positive Edge Detect)
8	9	3		P1.7	INT5 External Interrupt 5 (Negative Edge Detect)
21	24	18	A8 (P2.0)	<b>A8–15 (Port 2) – Output.</b> Port 2 serves as the MSB for external addressing. P2.7 is A15 and P2.0 is A8. The DS80C310 will automatically place the MSB of an address on P2 for external ROM and RAM access. Although Port 2 can be accessed like an ordinary I/O port, the value stored on the Port 2 latch will never be seen on the pins (due to memory access). Therefore writing to Port 2, in software is only useful for the instructions MOVX A, @ Ri or MOVX @ Ri, A. These instructions use the Port 2 internal latch to supply the external address MSB. In this case, the Port 2 latch value will be supplied as the address information.	
22	25	19	A9 (P2.1)		
23	26	20	A10 (P2.2)		
24	27	21	A11 (P2.3)		
25	28	22	A12 (P2.4)		
26	29	23	A13 (P2.5)		
27	30	24	A14 (P2.6)		
28	31	25	A15 (P2.7)		
10–17	11, 13–19	5, 7–13	P3.0–P3.7	<b>Port 3 – I/O.</b> Port 3 functions as both an 8-bit bi-directional I/O port and an alternate functional interface for external Interrupts, Serial Port 0, Timer 0 and 1 Inputs, RD and WR strobes. The reset condition of Port 3 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS80C310 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port once again becomes both the output high and input state. The alternate modes of Port 3 are outlined below.	
				Port	Alternate Mode
10	11	5		P3.0	RXD0 Serial Port 0 Input
11	13	7		P3.1	TXD0 Serial Port 0 Output
12	14	8		P3.2	INT0 External Interrupt 0
13	15	9		P3.3	INT1 External Interrupt 1
14	16	10		P3.4	T0 Timer 0 External Input
15	17	11		P3.5	T1 Timer 1 External Input
16	18	12		P3.6	WR External Data Memory Write Strobe
17	19	13		P3.7	RD External Data Memory Read Strobe
31	35	29	EA	<b>EA – Input.</b> This pin must be connected to ground for proper operation.	
–	12 34	6 28	NC	<b>NC – Reserved.</b> These pins should not be connected. They are reserved for use with future devices in this family.	

## COMPATIBILITY

The DS80C310 is a fully static CMOS 8051 compatible microcontroller designed for high performance. In most cases the DS80C310 can drop into an existing socket for the 80C31 or 80C32 to improve the operation significantly. In general, software written for existing 8051 based systems works without modification on the DS80C310. The exception is critical timing since the High-Speed Micro performs its instructions much faster than the original for any given crystal selection. The DS80C310 runs the standard 8051 family instruction set and is pin compatible with DIP, PLCC or TQFP packages. The DS80C310 is a streamlined version of the DS80C320. It maintains upward compatibility but has fewer peripherals.

The DS80C310 provides three 16-bit timer/counters, a full-duplex serial port, and 256 bytes of direct RAM. I/O ports have the same operation as a standard 8051 product. Timers will default to a 12 clock per cycle operation to keep their timing compatible with original 8051 family systems. However, timers are individually programmable to run at the new 4 clocks per cycle if desired.

The DS80C310 provides several new hardware functions that are controlled by Special Function registers. A summary of the Special Function Registers is provided in Table 2.

## PERFORMANCE OVERVIEW

The DS80C310 features a high-speed 8051 compatible core. Higher speed comes not just from increasing the clock frequency, but from a newer, more efficient design.

This updated core does not have the dummy memory cycles that are present in a standard 8051. A conventional 8051 generates machine cycles using the clock frequency divided by 12. In the DS80C310, the same machine cycle takes four clocks. Thus the fastest instruction, 1 machine cycle, executes three times faster for the same crystal frequency. Note that these are identical instructions. The majority of instructions on the DS80C310 will see the full 3 to 1 speed improvement. Some instructions will get between 1.5 and 2.4 to 1 improvement. All instructions are faster than the original 8051.

The numerical average of all opcodes gives approximately a 2.5 to 1 speed improvement. Improvement of

individual programs will depend on the actual instructions used. Speed sensitive applications would make the most use of instructions that are three times faster. However, the sheer number of 3 to 1 improved opcodes makes dramatic speed improvements likely for any code. These architecture improvements and 0.8  $\mu$ m CMOS produce a peak instruction cycle in 121 ns (8.25 MIPs). The Dual Data Pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

## INSTRUCTION SET SUMMARY

All instructions in the DS80C310 perform the same functions as their 8051 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute and relative number of clocks.

For absolute timing of real-time events, the timing of software loops can be calculated using a table in the High-Speed Microcontroller User's Guide. However, counter/timers default to run at the older 12 clocks per increment. In this way, timer-based events occur at the standard intervals with software executing at higher speed. Timers optionally can run at 4 clocks per increment to take advantage of faster processor operation.

The relative time of two instructions might be different in the new architecture than it was previously. For example, in the original architecture, the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction used two machine cycles or 24 oscillator cycles. Therefore, they required the same amount of time. In the DS80C310, the MOVX instruction takes as little as two machine cycles or eight oscillator cycles but the "MOV direct, direct" uses three machine cycles or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times. This is because the DS80C310 usually uses one instruction cycle for each instruction byte. The user concerned with precise program timing should examine the timing of each instruction for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV. Refer to the High-Speed Microcontroller User's Guide for details and individual instruction timing.

## SPECIAL FUNCTION REGISTERS

Special Function Registers (SFRs) control most special features of the DS80C310. The High-Speed Microcontroller User's Guide describes all SFRs. Functions that are not part of the standard 80C32 are in bold.

**SPECIAL FUNCTION REGISTERS** Table 2

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
<b>DPL1</b>									84h
<b>DPH1</b>									85h
<b>DPS</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>SEL</b>	86h
PCON	SMOD	SMOD0	—	—	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
<b>CKCON</b>	<b>—</b>	<b>—</b>	<b>T2M</b>	<b>T1M</b>	<b>T0M</b>	<b>MD2</b>	<b>MD1</b>	<b>MD0</b>	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
<b>EXIF</b>	<b>IE5</b>	<b>IE4</b>	<b>IE3</b>	<b>IE2</b>	—	—	—	—	91h
SCON	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	98h
SBUF									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	—	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	—	—	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
STATUS	0	HIP	LIP	1	1	1	1	1	C5h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	—	—	—	—	—	—	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
<b>WDCON</b>	<b>—</b>	<b>POR</b>	—	—	—	—	—	—	D8h
ACC									E0h
<b>EIE</b>	<b>—</b>	<b>—</b>	<b>—</b>	<b>—</b>	<b>EX5</b>	<b>EX4</b>	<b>EX3</b>	<b>EX2</b>	E8h
B									F0h
<b>EIP</b>	<b>—</b>	<b>—</b>	<b>—</b>	<b>—</b>	<b>PX5</b>	<b>PX4</b>	<b>PX3</b>	<b>PX2</b>	F8h

### MEMORY ACCESS

The DS80C310 contains no on-chip ROM, and 256 bytes of scratchpad RAM. Off-chip memory is accessed using the multiplexed address/data bus on P0 and the MSB address on P2. Timing diagrams are provided in the Electrical Specifications. Program memory (ROM) is accessed at a fixed rate determined by the crystal frequency and the actual instructions. As mentioned above, an instruction cycle requires four clocks. Data memory (RAM) is accessed according to a variable speed MOVX instruction as described below.

### STRETCH MEMORY CYCLE

The DS80C310 allows the application software to adjust the speed of data memory access. The micro is capable of performing the MOVX in as few as two instruction cycles. However, this value can be stretched as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic. Even in high-speed systems, it may not be necessary or desirable to perform data memory access at full speed. In addition, there are a variety of memory mapped peripherals such as LCD displays or UARTs that are not fast.

The Stretch MOVX is controlled by the Clock Control Register at SFR location 8Eh as described below. This allows the user to select a stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. A Stretch of seven will result in a MOVX of

nine machine cycles. Software can dynamically change this value depending on the particular memory or peripheral.

On reset, the Stretch value will default to a one resulting in a three cycle MOVX. Therefore, RAM access will not be performed at full speed. This is a convenience to existing designs that may not have fast RAM in place. When maximum speed is desired, the software should select a Stretch value of zero. When using very slow RAM or peripherals, a larger stretch value can be selected. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven causes the microcontroller to stretch the read/write strobe and all related timing. This results in a wider read/write strobe allowing more time for memory/peripherals to respond. The timing of the variable speed MOVX is shown in the Electrical Specifications. Note that full speed access is not the reset default case. Table 3 shows the resulting strobe widths for each Stretch value. The memory stretch is implemented using the Clock Control Special Function Register at SFR location 8Eh. The stretch value is selected using bits CKCON.2–0. In the table, these bits are referred to as M2 through M0. The first stretch (default) allows the use of common 120 ns or 150 ns RAMs without dramatically lengthening the memory access.

DATA MEMORY CYCLE STRETCH VALUES Table 3

CKCON.2–0			MEMORY CYCLES	RD OR WR STROBE WIDTH IN CLOCKS	STROBE WIDTH TIME	
M2	M1	M0			@ 25 MHz	@ 33 MHz
0	0	0	2	2	80 ns	60 ns
0	0	1	3 (default)	4	160 ns	121 ns
0	1	0	4	8	320 ns	242 ns
0	1	1	5	12	480 ns	364 ns
1	0	0	6	16	640 ns	485 ns
1	0	1	7	20	800 ns	606 ns
1	1	0	8	24	960 ns	727 ns
1	1	1	9	28	1120 ns	848 ns



## DUAL DATA POINTER

Data memory block moves can be accelerated using the DS80C310 Dual Data Pointer (DPTR). The standard 8032 DPTR is a 16-bit value that is used to address off-chip data RAM or peripherals. In the DS80C310, the standard data pointer is called DPTR and is located at SFR addresses 82h and 83h. These are the standard locations. No modification of standard code is needed to use DPTR. The new DPTR is located at SFR 84h and 85h and is called DPTR1. The DPTR Select bit (DPS) chooses the active pointer and is located at the lsb of the SFR location 86h. No other bits in register 86h have any effect and are set to 0. The user switches between data pointers by toggling the lsb of register 86h. The increment (INC) instruction is the fastest way to accomplish this. All DPTR-related instructions use the currently selected DPTR for any activity. Therefore only one instruction is required to switch from a source to a destination address. Using the Dual Data Pointer saves code from needing to save source and destination addresses when doing a block move. Once loaded, the software simply switches between DPTR0 and 1. The relevant register locations are as follows.

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (lsb)

## STOP MODE ENHANCEMENTS

Setting bit 1 of the Power Control register (PCON; 87h) invokes the Stop mode. Stop mode is the lowest power state since it turns off all internal clocking. The  $I_{CC}$  of a standard Stop mode is approximately 1  $\mu$ A (but is specified in the Electrical Specifications). The CPU will exit Stop mode from an external interrupt or a reset condition. Internally generated interrupts are not useful since they require clocking activity.

The DS80C310 allows a resume from Stop using a INT2–5, which are edge triggered interrupts. The start-up timing is managed by an internal crystal counter. A delay of 65,536 clocks occurs to give the crystal enough time to start and stabilize.

## PERIPHERAL OVERVIEW

The DS80C310 provides the same peripheral functions as the standard 80C32. It is compatible with the DS80C320 but does not offer all of the peripherals.

## TIMER RATE CONTROL

There is one important difference between the DS80C310 and 8051 regarding timers. The original 8051 used 12 clocks per cycle for timers as well as for machine cycles. The DS80C310 architecture normally uses 4 clocks per machine cycle. However, in the area of timers and serial ports, the DS80C310 will default to 12 clocks per cycle on reset. This allows existing code with real-time dependencies such as baud rates to operate properly.

If an application needs higher speed timers or serial baud rates, the user can select individual timers to run at the 4 clock rate. The Clock Control register (CKCON; 8Eh) determines these timer speeds. When the relevant CKCON bit is a logic 1, the DS80C310 uses 4 clocks per cycle to generate timer speeds. When the bit is a 0, the DS80C310 uses 12 clocks for timer speeds. The reset condition is a 0. CKCON.5 selects the speed of Timer 2. CKCON.4 selects Timer 1 and CKCON.3 selects Timer 0. Note that unless a user desires very fast timing, it is unnecessary to alter these bits. Note that the timer controls are independent.

## POWER ON RESET

The DS80C310 will hold itself in reset during a power up until 65,536 clock cycles have elapsed. The power-on reset used by the DS80C310 differs somewhat from other members of the High-Speed Microcontroller family. The crystal oscillator may start anywhere between 1.0V and 4.5V, but is not specified. This eliminates the need for an RC reset circuit. For voltage specific precision brownout detection, an external component will be needed. When the device goes through a power on reset, the POR flag will be set in the WDCON (D8h) register at bit 6.

## INTERRUPTS

The DS80C310 provides 10 interrupt sources with two priority levels. Software can assign high or low priority to all sources. All interrupts that are new to the 8051 have a lower natural priority than the originals.



INTERRUPT SOURCES AND PRIORITIES Table 4

NAME	DESCRIPTION	VECTOR	NATURAL PRIORITY
INT0	External Interrupt 0	03h	1
TF0	Timer 0	0Bh	2
INT1	External Interrupt 1	13h	3
TF1	Timer 1	1Bh	4
SCON	TI or RI from the serial port	23h	5
TF2	Timer 2	2Bh	6
INT2	External Interrupt 2	43h	7
INT3	External Interrupt 3	4Bh	8
INT4	External Interrupt 4	53h	9
INT5	External Interrupt 5	5Bh	10

**POWER ON RESET**

The DS80C310 will not start in reset during a power-up until 65,536 clock cycles have elapsed. The power-on reset used by the DS80C310 differs somewhat from other members of the High-Speed Microcontroller family. The crystal oscillator may start anywhere between 1.0V and 1.5V but is not specified. This eliminates the need for an RC reset circuit. For voltage specific precision brownout detection, an external component will be needed. When the device goes through a power-on reset, the POR flag will be set in the WDCON (03h) register at bit 5.

**INTERRUPTS**

The DS80C310 provides 10 interrupt sources with two priority levels. Software can assign high or low priority to all sources. All interrupts that are new to the DS80C310 have a lower natural priority than the original.

DPL	82h	Low byte original DPTA
DPH	83h	High byte original DPTA
DPL1	92h	Low byte new DPTA
DPH1	93h	High byte new DPTA
DPS	86h	DPTA Select (ab)

**STOP MODE ENHANCEMENTS**

Setting bit 1 of the Power Control register (PCON: 87h) invokes the Stop mode. Stop mode is the lowest power state since it turns off all internal clocking. The I/O of a standard Stop mode is approximately 1  $\mu$ A (but is specified in the Electrical Specifications). The CPU will exit Stop mode from an external interrupt or a reset condition. Internally generated interrupts are not useful since they require clocking activity.

The DS80C310 allows a resume from Stop using a INT2-5, which are edge triggered interrupts. The start-up timing is managed by an internal on-chip counter. A delay of 65,536 clock cycles to give the crystal enough time to start and stabilize.

**PERIPHERAL OVERVIEW**

The DS80C310 provides the same peripheral functions as the standard 80C32. It is compatible with the DS80C320 but does not offer all of the peripheral.

**ABSOLUTE MAXIMUM RATINGS\***

Voltage on Any Pin Relative to Ground

-0.3V to +7.0V

Operating Temperature

0°C to 70°C

Storage Temperature

-55°C to +125°C

Soldering Temperature

260°C for 10 seconds

\* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

**DC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}$ =4.0V to 5.5V)

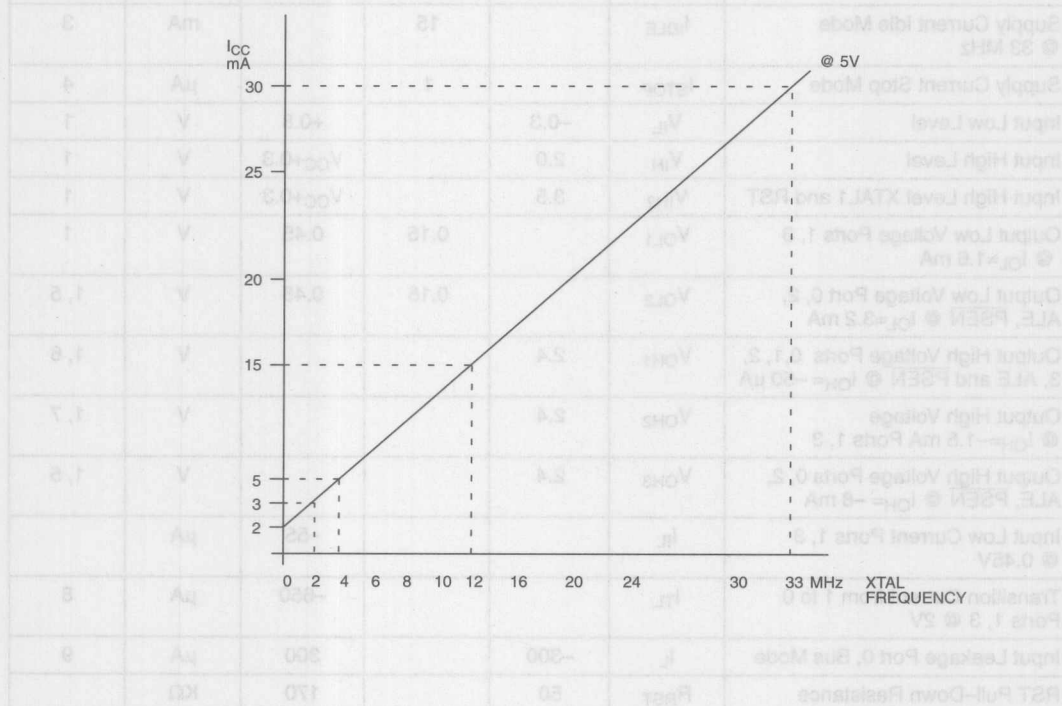
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	$V_{CC}$	4.0	5.0	5.5	V	1
Supply Current Active Mode @ 33 MHz	$I_{CC}$		30		mA	2
Supply Current Idle Mode @ 33 MHz	$I_{IDLE}$		15		mA	3
Supply Current Stop Mode	$I_{STOP}$		1		$\mu$ A	4
Input Low Level	$V_{IL}$	-0.3		+0.8	V	1
Input High Level	$V_{IH}$	2.0		$V_{CC}+0.3$	V	1
Input High Level XTAL1 and RST	$V_{IH2}$	3.5		$V_{CC}+0.3$	V	1
Output Low Voltage Ports 1, 3 @ $I_{OL}=1.6$ mA	$V_{OL1}$		0.15	0.45	V	1
Output Low Voltage Port 0, 2, ALE, PSEN @ $I_{OL}=3.2$ mA	$V_{OL2}$		0.15	0.45	V	1, 5
Output High Voltage Ports 0, 1, 2, 3, ALE and PSEN @ $I_{OH}=-50$ $\mu$ A	$V_{OH1}$	2.4			V	1, 6
Output High Voltage @ $I_{OH}=-1.5$ mA Ports 1, 3	$V_{OH2}$	2.4			V	1, 7
Output High Voltage Ports 0, 2, ALE, PSEN @ $I_{OH}=-8$ mA	$V_{OH3}$	2.4			V	1, 5
Input Low Current Ports 1, 3 @ 0.45V	$I_{IL}$			-55	$\mu$ A	
Transition Current from 1 to 0 Ports 1, 3 @ 2V	$I_{TL}$			-650	$\mu$ A	8
Input Leakage Port 0, Bus Mode	$I_L$	-300		300	$\mu$ A	9
RST Pull-Down Resistance	$R_{RST}$	50		170	K $\Omega$	

**NOTES FOR DC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature operation unless otherwise noted.

1. All voltages are referenced to ground.
2. Active current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=RST=5.5$ V, all other pins disconnected.
3. Idle mode current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=5.5$ V, RST at ground, all other pins disconnected.

4. Stop mode current measured with XTAL1 and RST grounded,  $V_{CC}=5.5V$ , all other pins disconnected.
5. When addressing external memory.
6.  $RST=5.5V$ . This condition mimics operation of pins in I/O mode.
7. During a 0 to 1 transition, a one-shot drives the ports hard for two clock cycles. This measurement reflects port in transition mode.
8. Ports 1, 2, and 3 source transition current when being pulled down externally. It reaches its maximum at approximately 2V.
9.  $0.45 < V_{IN} < V_{CC}$ . Not a high impedance input. This port is a weak address holding latch because Port 0 is dedicated as an address bus on the DS80C320. Peak current occurs near the input transition point of the latch, approximately 2V.

TYPICAL  $I_{CC}$  VERSUS FREQUENCY Figure 2

## NOTES FOR DC ELECTRICAL CHARACTERISTICS

1. All parameters apply to both commercial and industrial temperature operation unless otherwise noted.
2. Active current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=5.5V$ ,  $RST=5.5V$ , all other pins disconnected.
3. Idle mode current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=5.5V$ ,  $RST$  at ground, all other pins disconnected.

**AC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	25 MHz		VARIABLE CLOCK		NOTES
		MIN	MAX	MIN	MAX	
Oscillator Frequency	$1/t_{CLCL}$	0	33	0	33	MHz
ALE Pulse Width	$t_{LHLL}$	40		$1.5t_{CLCL}-5$		ns
Port 0 Address Valid to ALE Low	$t_{AVLL}$	10		$0.5t_{CLCL}-5$		ns
Address Hold after ALE Low	$t_{LLAX}$	10		$0.5t_{CLCL}-5$		ns
ALE Low to Valid Instruction In	$t_{LLIV}$		56		$2.5t_{CLCL}-20$	ns
ALE Low to $\overline{PSEN}$ Low	$t_{LLPL}$	10		$0.5t_{CLCL}-5$		ns
$\overline{PSEN}$ Pulse Width	$t_{PLPH}$	55		$2t_{CLCL}-5$		ns
$\overline{PSEN}$ Low to Valid Instr. In	$t_{PLIV}$		41		$2t_{CLCL}-20$	ns
Input Instruction Hold after $\overline{PSEN}$	$t_{PXIX}$	0		0		ns
Input Instruction Float after $\overline{PSEN}$	$t_{PXIZ}$		26		$t_{CLCL}-5$	ns
Port 0 Address to Valid Instr. In	$t_{AVIV1}$		71		$3t_{CLCL}-20$	ns
Port 2 Address to Valid Instr. In	$t_{AVIV2}$		81		$3.5t_{CLCL}-25$	ns
$\overline{PSEN}$ Low to Address Float	$t_{PLAZ}$		0		0	ns

**NOTES FOR AC ELECTRICAL CHARACTERISTICS**

All parameters apply to both commercial and industrial temperature range operation unless otherwise noted. All signals characterized with load capacitance of 80 pF except Port 0, ALE,  $\overline{PSEN}$ ,  $\overline{RD}$  and  $\overline{WR}$  with 100 pF. Interfacing to memory devices with float times (turn off times) over 25 ns may cause contention. This will not damage the parts, but will cause an increase in operating current.

MS	MT	MB	MOVX CYCLES	Access
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	0
0	1	0	4 machine cycles	0
0	1	1	5 machine cycles	0
1	0	0	6 machine cycles	1
1	0	1	7 machine cycles	1
1	1	0	8 machine cycles	1
1	1	1	9 machine cycles	1

## MOVX CHARACTERISTICS

(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	VARIABLE CLOCK		UNITS	STRETCH
		MIN	MAX		
Data Access ALE Pulse Width	$t_{LLHL2}$	$1.5t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Address Hold after ALE Low for MOVX Write	$t_{LLAX2}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Pulse Width	$t_{RLRH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{WR}$ Pulse Width	$t_{WLWH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Valid Data In	$t_{RLDV}$		$2t_{CLCL}-20$ $t_{MCS}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold after Read	$t_{RHDX}$	0		ns	
Data Float after Read	$t_{RHDZ}$		$t_{CLCL}-5$ $2t_{CLCL}-5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to Valid Data In	$t_{LLDV}$		$2.5t_{CLCL}-20$ $t_{CLCL}+t_{MCS}-40$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to Valid Data In	$t_{AVDV1}$		$3t_{CLCL}-20$ $1.5t_{CLCL}+t_{MCS}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to Valid Data In	$t_{AVDV2}$		$3.5t_{CLCL}-20$ $2t_{CLCL}+t_{MCS}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$t_{LLWL}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$	$0.5t_{CLCL}+5$ $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL1}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL2}$	$1.5t_{CLCL}-10$ $2.5t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Valid to $\overline{WR}$ Transition	$t_{QVWX}$	-5		ns	$t_{MCS}=0$
Data Hold after Write	$t_{WHQX}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Address Float	$t_{RLAZ}$		$-0.5t_{CLCL}-5$	ns	
$\overline{RD}$ or $\overline{WR}$ High to ALE High	$t_{WHLH}$	0 $t_{CLCL}-5$	10 $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$

NOTE:  $t_{MCS}$  is a time period related to the Stretch memory cycle selection. The following table shows the value of  $t_{MCS}$  for each Stretch selection.

M2	M1	M0	MOVX CYCLES	$t_{MCS}$
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	4 $t_{CLCL}$
0	1	0	4 machine cycles	8 $t_{CLCL}$
0	1	1	5 machine cycles	12 $t_{CLCL}$
1	0	0	6 machine cycles	16 $t_{CLCL}$
1	0	1	7 machine cycles	20 $t_{CLCL}$
1	1	0	8 machine cycles	24 $t_{CLCL}$
1	1	1	9 machine cycles	28 $t_{CLCL}$



**EXTERNAL CLOCK CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Clock High Time	$t_{CHCX}$	10			ns	
Clock Low Time	$t_{CLCX}$	10			ns	
Clock Rise Time	$t_{CLCL}$			5	ns	
Clock Fall Time	$t_{CHCL}$			5	ns	

**SERIAL PORT MODE 0 TIMING CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

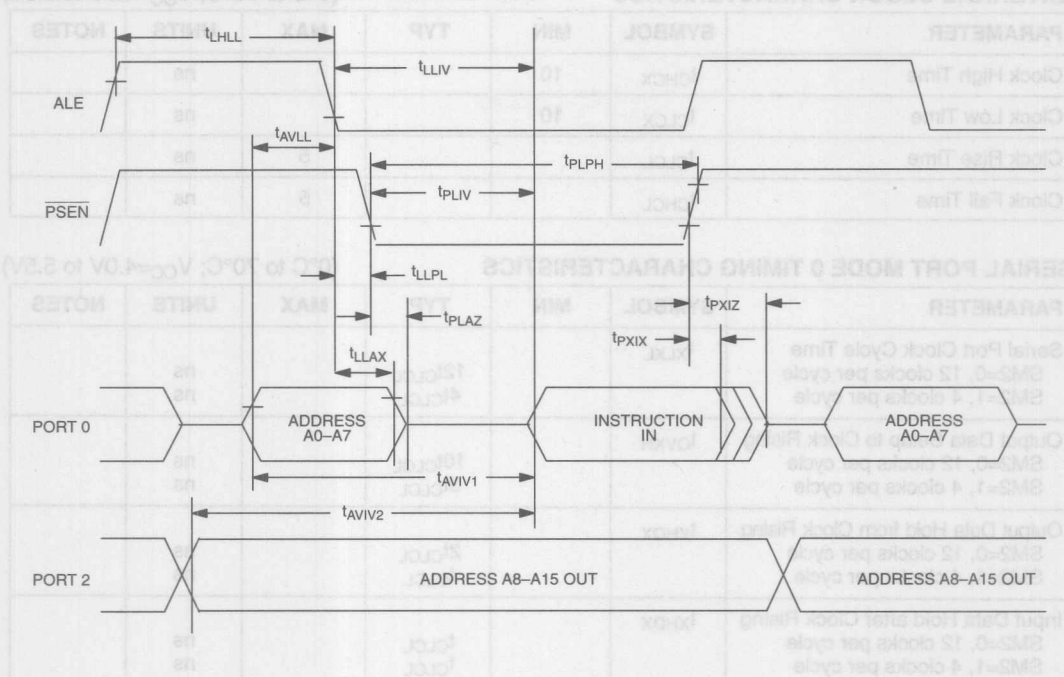
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Serial Port Clock Cycle Time SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{LXL}$		$12t_{CLCL}$ $4t_{CLCL}$		ns ns	
Output Data Setup to Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{QVXH}$		$10t_{CLCL}$ $3t_{CLCL}$		ns ns	
Output Data Hold from Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHGX}$		$2t_{CLCL}$ $t_{CLCL}$		ns ns	
Input Data Hold after Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHDX}$		$t_{CLCL}$ $t_{CLCL}$		ns ns	
Clock Rising Edge to Input Data Valid SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHDV}$		$11t_{CLCL}$ $3t_{CLCL}$		ns ns	

**EXPLANATION OF AC SYMBOLS**

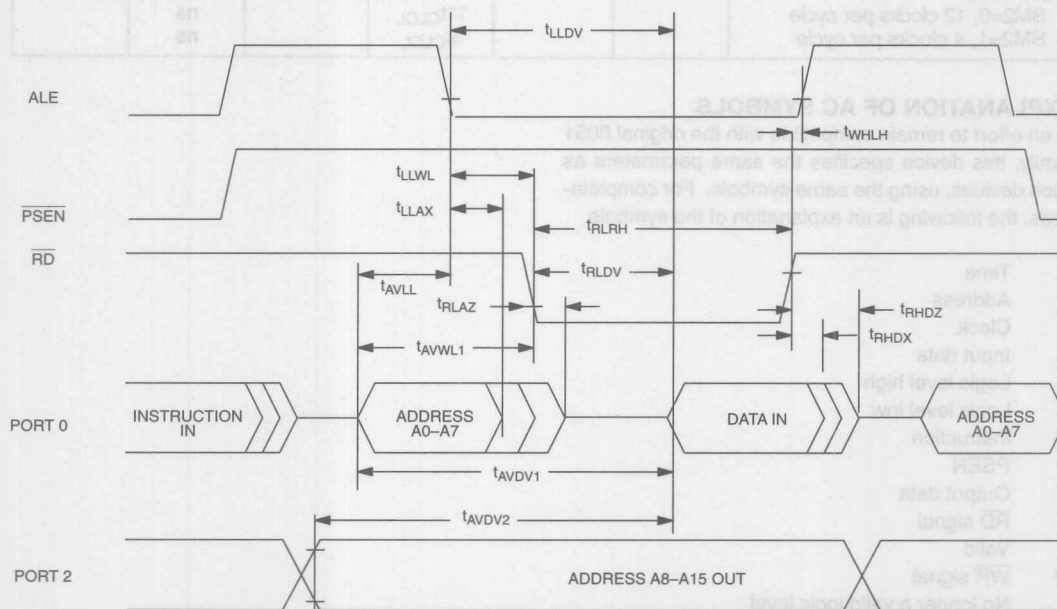
In an effort to remain compatible with the original 8051 family, this device specifies the same parameters as such devices, using the same symbols. For completeness, the following is an explanation of the symbols.

t	Time
A	Address
C	Clock
D	Input data
H	Logic level high
L	Logic level low
I	Instruction
P	PSEN
Q	Output data
R	$\overline{RD}$ signal
V	Valid
W	$\overline{WR}$ signal
X	No longer a valid logic level
Z	Tristate

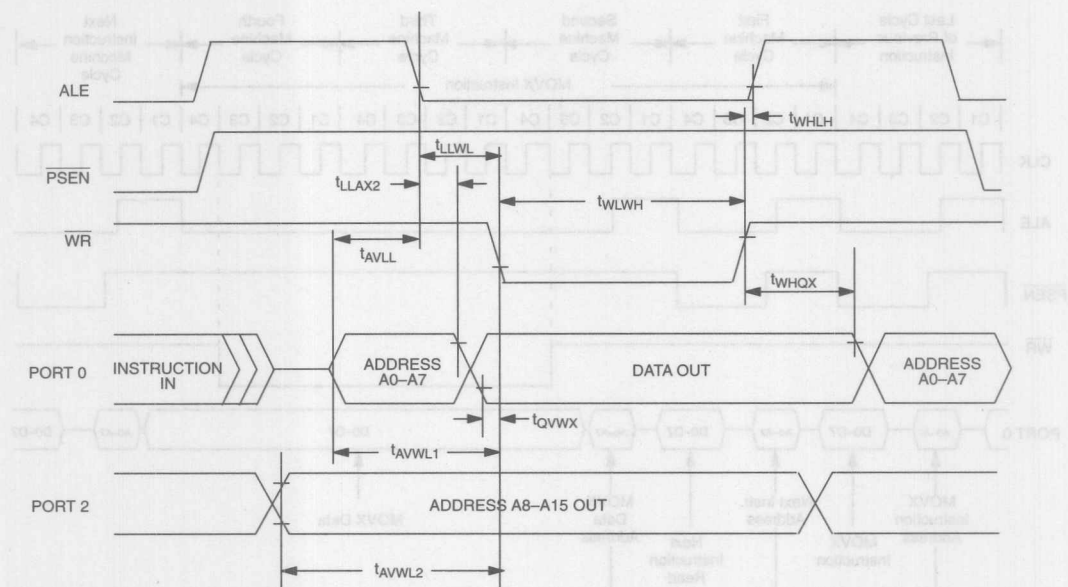
## EXTERNAL PROGRAM MEMORY READ CYCLE



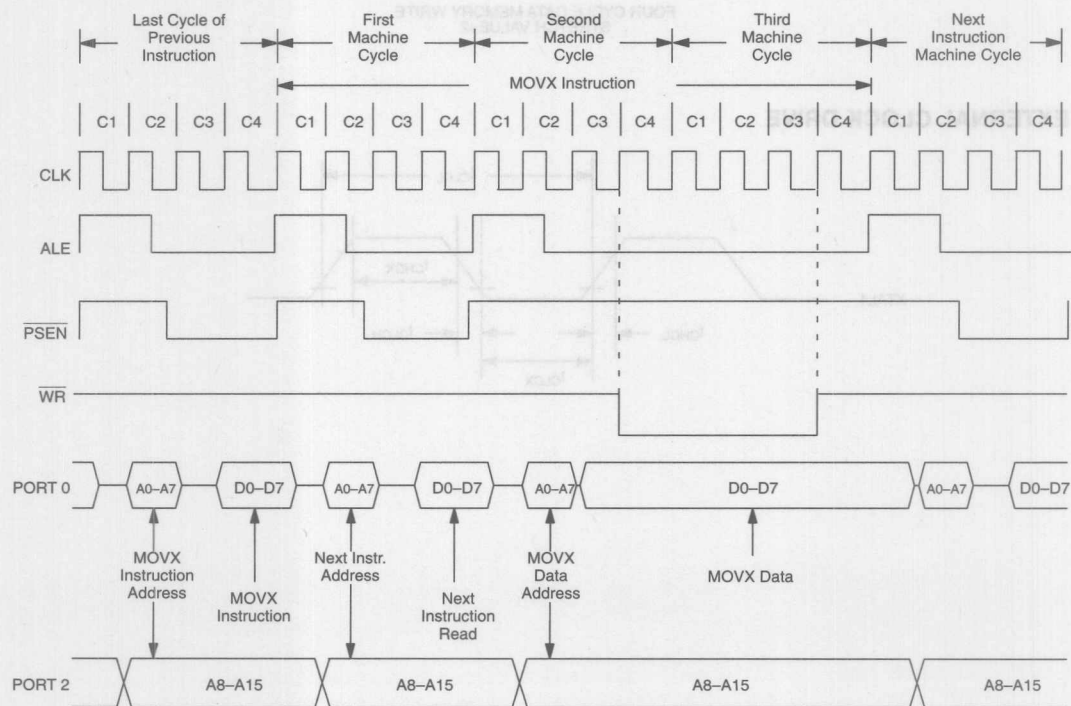
## EXTERNAL DATA MEMORY READ CYCLE



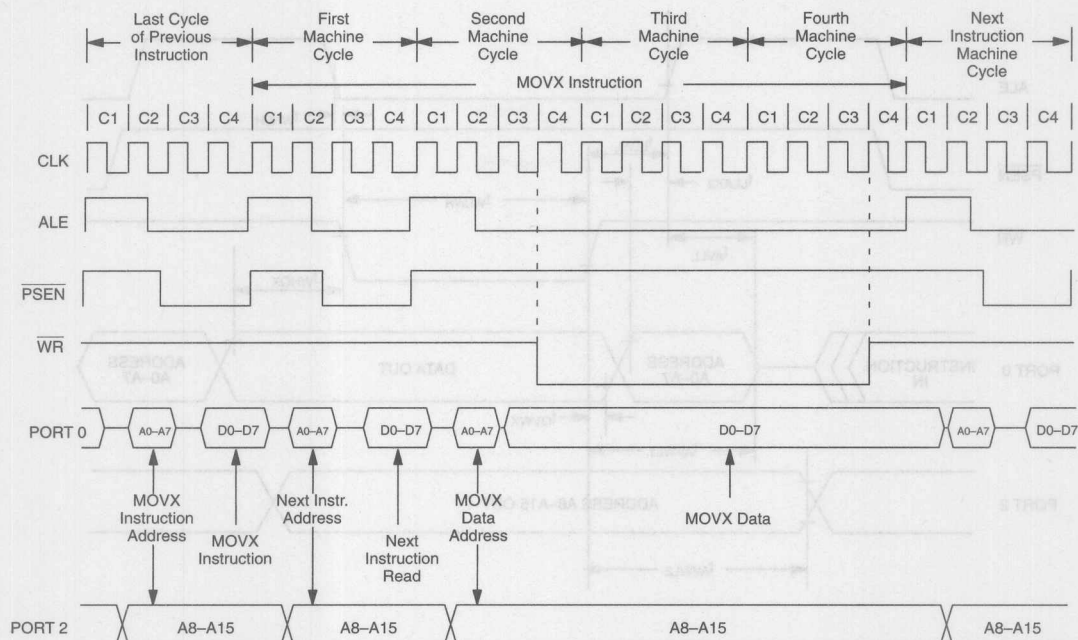
## DATA MEMORY WRITE CYCLE



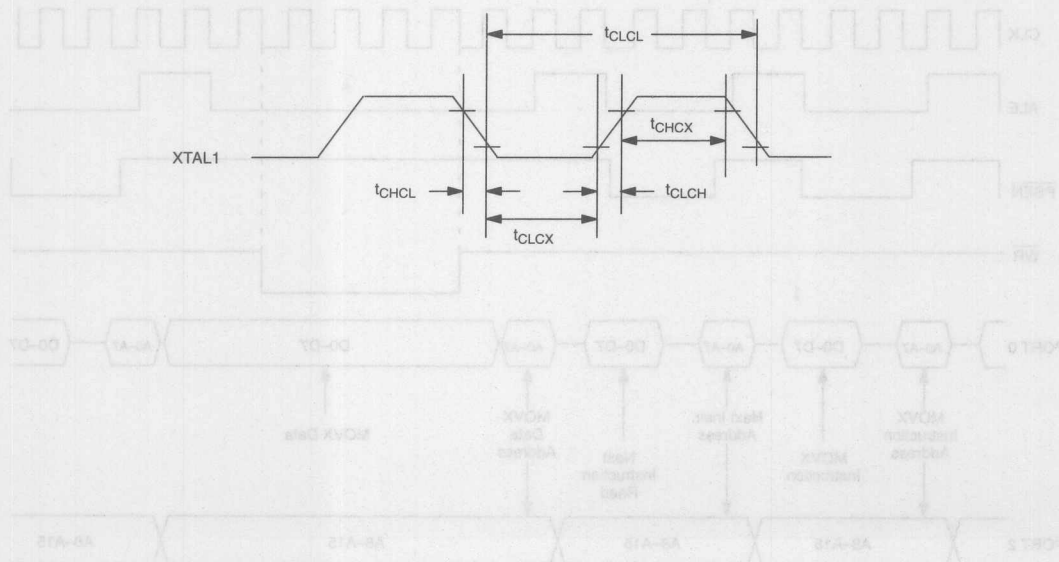
## DATA MEMORY WRITE WITH STRETCH=1



## DATA MEMORY WRITE WITH STRETCH=2

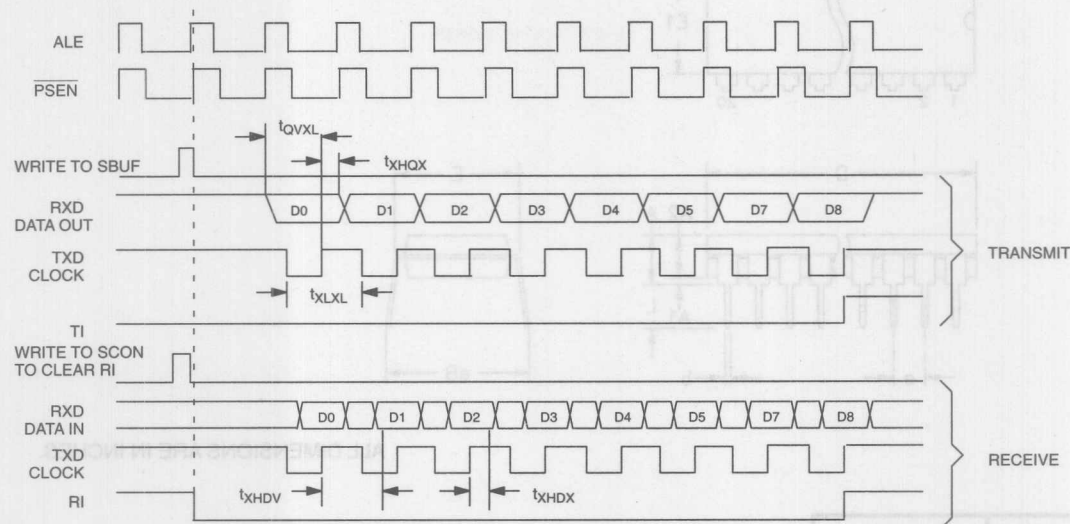
FOUR CYCLE DATA MEMORY WRITE  
STRETCH VALUE=2

## EXTERNAL CLOCK DRIVE

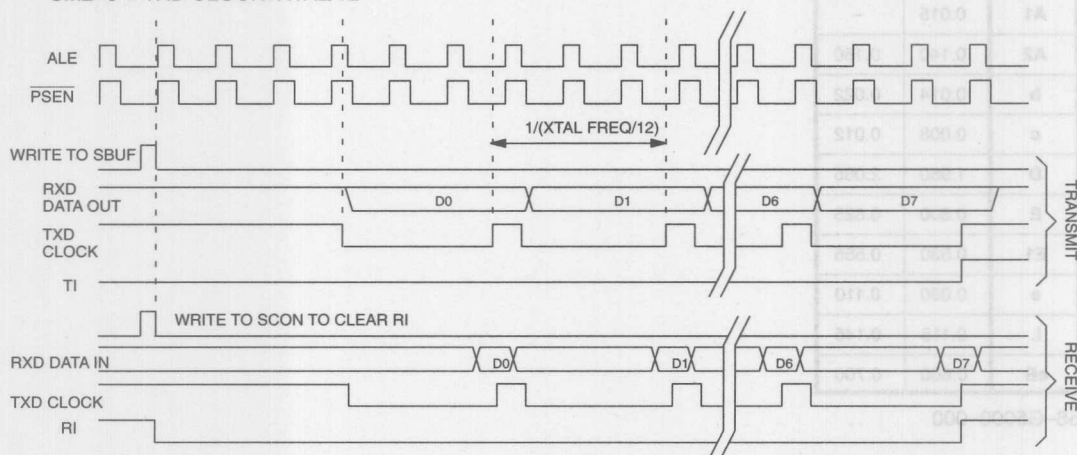


## SERIAL PORT MODE 0 TIMING

SERIAL PORT 0 (SYNCHRONOUS MODE)  
HIGH SPEED OPERATION SM2=1=>TXD CLOCK=XTAL/4



SERIAL PORT 0 (SYNCHRONOUS MODE)  
SM2=0=>TXD CLOCK=XTAL/12



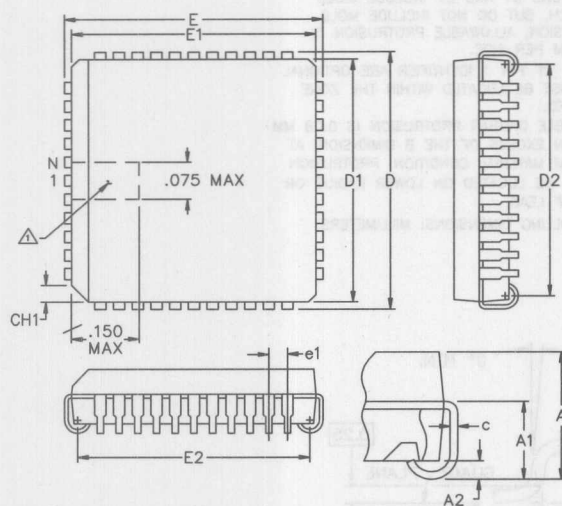




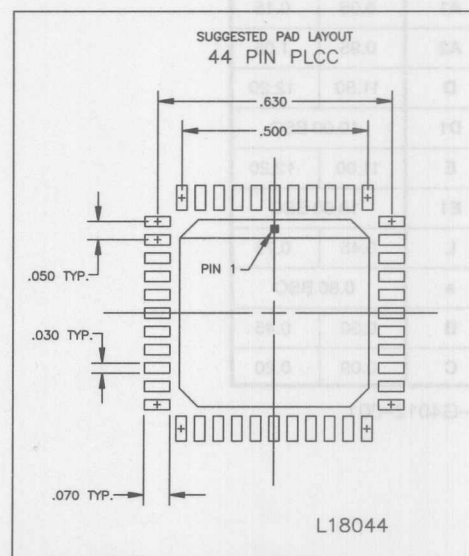
## 44-PIN PLCC

## NOTE:

1.  $\Delta$  PIN-1 IDENTIFIER TO BE LOCATED IN ZONE INDICATED.
2. CONTROLLING DIMENSIONS ARE IN INCHS

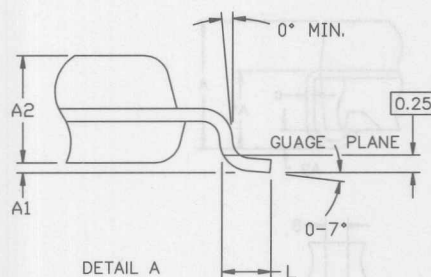
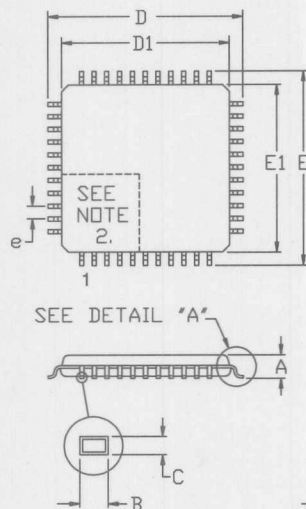


PKG	44-PIN	
DIM	MIN	MAX
A	0.165	0.180
A1	0.090	0.120
A2	0.020	—
B	0.026	0.033
B1	0.013	0.021
c	0.009	0.012
CH1	0.042	0.048
D	0.685	0.695
D1	0.650	0.656
D2	0.590	0.630
E	0.685	0.695
E1	0.650	0.656
E2	0.590	0.630
e1	0.050 BSC	
N	44	—



56-G4003-001

## 44-PIN TQFP

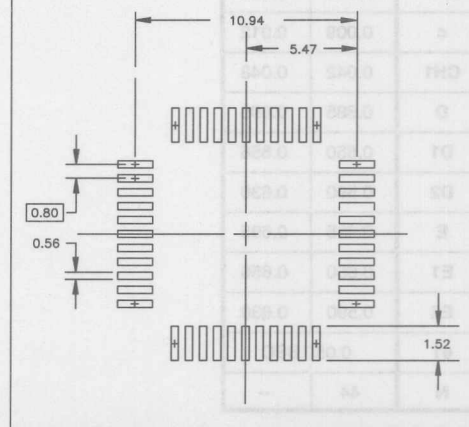


PKG	44-PIN	
DIM	MIN	MAX
A	—	1.20
A1	0.05	0.15
A2	0.95	1.05
D	11.80	12.20
D1	10.00 BSC	
E	11.80	12.20
E1	10.00 BSC	
L	0.45	0.75
e	0.80 BSC	
B	0.30	0.45
C	0.09	0.20

56-G4012-001

## NOTES:

1. DIMENSIONS D1 AND E1 INCLUDE MOLD MISMATCH, BUT DO NOT INCLUDE MOLD PROTRUSION; ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE.
2. DETAILS OF PIN 1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATED.
3. ALLOWABLE DAMBAR PROTRUSION IS 0.08 MM TOTAL IN EXCESS OF THE B DIMENSION; AT MAXIMUM MATERIAL CONDITION, PROTRUSION NOT TO BE LOCATED ON LOWER RADIUS OR FOOT OF LEAD.
4. CONTROLLING DIMENSIONS: MILLIMETERS.

SUGGESTED PAD LAYOUT  
44 PIN TQFP, 10\*10\*1.0

# DS80C320

## High-Speed Micro

# DALLAS

## SEMICONDUCTOR

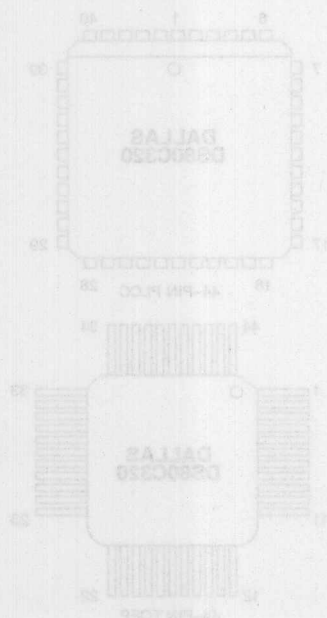
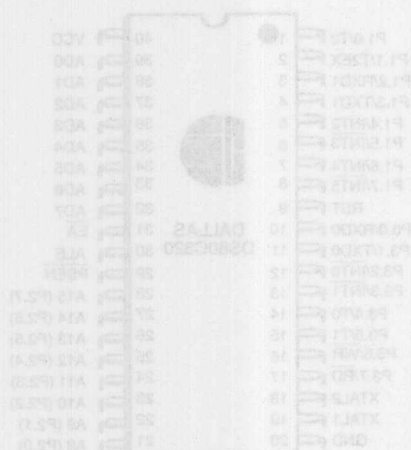
### FEATURES

- 80C32-Compatible
  - Pin-compatible
  - Standard 8051 instruction set
  - Four 8-bit I/O ports
  - Three 16-bit timers/counters
  - 256 bytes scratchpad RAM
  - Multiplexed address/data bus
  - Addresses 64KB ROM and 64KB RAM
- High-speed architecture
  - 4 clock-machine cycle (80C32=12)
  - Wasted cycles removed
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 151 ns
  - Uses less power for equivalent work
  - Dual data pointer
  - Optional variable length MOVX to access fast
  - slow RAM peripherals
- High integration controller includes:
  - Power-fail reset
  - Programmable Watchdog timer
  - Early-warning power-fail interrupt
- Two full-duplex hardware serial ports
- 13 total interrupt sources with six external
- Available in 40-pin DIP, 44-pin PLCC and TQFP

### DESCRIPTION

The DS80C320 is a fast 80C31/80C32-compatible microcontroller. Wasted clock and memory cycles have been removed using a redesigned processor core. As a result, every 8051 instruction is executed between 1.5 and 3 times faster than the original for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and same crystal. The DS80C320 offers a maximum crystal rate of 33 MHz, resulting in apparent execution speeds of 85.5 MHz (approximately 2.5X).

### PIN ASSIGNMENT



# DALLAS

SEMICONDUCTOR

## DS80C320

### High-Speed Micro

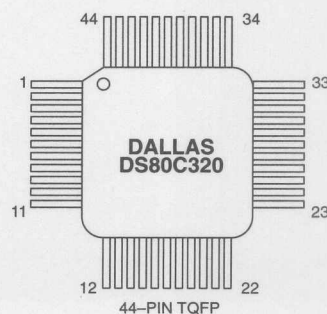
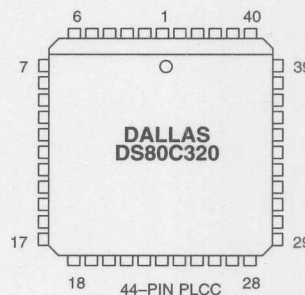
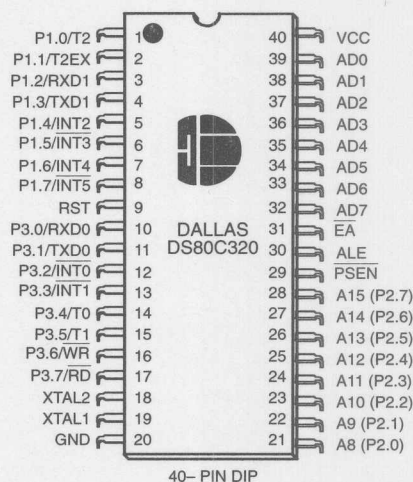
#### FEATURES

- 80C32-Compatible
  - Pin-compatible
  - Standard 8051 instruction set
  - Four 8-bit I/O ports
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
  - Multiplexed address/data bus
  - Addresses 64KB ROM and 64KB RAM
- High-speed architecture
  - 4 clocks/machine cycle (8032=12)
  - Wasted cycles removed
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Uses less power for equivalent work
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM/peripherals
- High integration controller includes:
  - Power-fail reset
  - Programmable Watchdog timer
  - Early-warning power-fail interrupt
- Two full-duplex hardware serial ports
- 13 total interrupt sources with six external
- Available in 40-pin DIP, 44-pin PLCC and TQFP

#### DESCRIPTION

The DS80C320 is a fast 80C31/80C32-compatible microcontroller. Wasted clock and memory cycles have been removed using a redesigned processor core. As a result, every 8051 instruction is executed between 1.5 and 3 times faster than the original for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and same crystal. The DS80C320 offers a maximum crystal rate of 33 MHz, resulting in apparent execution speeds of 82.5 MHz (approximately 2.5X).

#### PIN ASSIGNMENT





The DS80C320 is pin compatible with all three packages of the standard 80C32 and offers the same timer/counters, serial port, and I/O ports. In short, the DS80C320 is extremely familiar to 8051 users but provides the speed of a 16-bit processor.

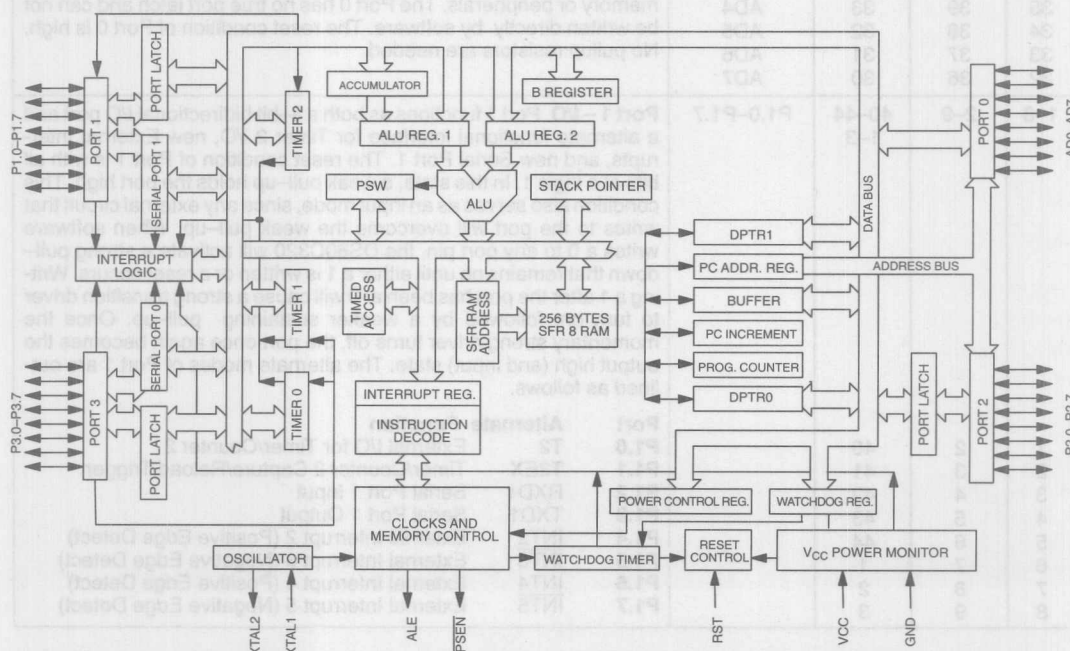
The DS80C320 provides several extras in addition to greater speed. These include a second full hardware

serial port, seven additional interrupts, programmable watchdog timer, power-fail interrupt and reset. The DS80C320 also provides dual data pointers (DPTRs) to speed block data memory moves. It can also adjust the speed of off-chip data memory access to between two and nine machine cycles for flexibility in selecting memory and peripherals.

## ORDERING INFORMATION

PART NUMBER	PACKAGE	MAX CLOCK SPEED	TEMPERATURE RANGE
DS80C320-MCG	40-pin plastic DIP	25 MHz	0°C to +70°C
DS80C320-QCG	44-pin PLCC	25 MHz	0°C to +70°C
DS80C320-ECG	44-pin TQFP	25 MHz	0°C to +70°C
DS80C320-MNG	40-pin plastic DIP	25 MHz	-40°C to +85°C
DS80C320-QNG	44-pin PLCC	25 MHz	-40°C to +85°C
DS80C320-ENG	44-pin TQFP	25 MHz	-40°C to +85°C
DS80C320-MCL	40-pin plastic DIP	33 MHz	0°C to +70°C
DS80C320-QCL	44-pin PLCC	33 MHz	0°C to +70°C
DS80C320-ECL	44-pin TQFP	33 MHz	0°C to +70°C
DS80C320-MNL	40-pin plastic DIP	33 MHz	-40°C to +85°C
DS80C320-QNL	44-pin PLCC	33 MHz	-40°C to +85°C
DS80C320-ENL	44-pin TQFP	33 MHz	-40°C to +85°C

DS80C320 BLOCK DIAGRAM Figure 1



PIN DESCRIPTION Table 1

DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION																		
40	44	38	V <sub>CC</sub>	V <sub>CC</sub> – +5V.																		
20	22, 23	16, 17	GND	GND – Digital circuit ground.																		
9	10	4	RST	<b>RST – Input.</b> The RST input pin contains a schmitt voltage input to recognize external active high Reset inputs. The pin also employs an internal pull-down resistor to allow for a combination of wired OR external Reset sources. An RC is <u>not</u> required for power-up, as the DS80C320 provides this function internally.																		
18 19	20 21	14 15	XTAL2 XTAL1	<b>XTAL1, XTAL2</b> – The crystal oscillator pins XTAL1 and XTAL2 provide support for parallel resonant, AT cut crystals. XTAL1 acts also as an input in the event that an external clock source is used in place of a crystal. XTAL2 serves as the output of the crystal amplifier.																		
29	32	26	PSEN	<b>PSEN – Output.</b> The Program Store Enable output. This signal is commonly connected to external ROM memory as a chip enable. PSEN will provide an active low pulse width of 2.25 XTAL1 cycles with a period of four XTAL1 cycles. PSEN is driven high when data memory (RAM) is being accessed through the bus and during a reset condition.																		
30	33	27	ALE	<b>ALE – Output.</b> The Address Latch Enable output functions as a clock to latch the external address LSB from the multiplexed address/data bus. This signal is commonly connected to the latch enable of an external 373 family transparent latch. ALE has a pulse width of 1.5 XTAL1 cycles and a period of four XTAL1 cycles. ALE is forced high when the DS80C320 is in a Reset condition.																		
39 38 37 36 35 34 33 32	43 42 41 40 39 38 37 36	37 36 35 34 33 32 31 30	AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7	<b>AD0–7 (Port 0) – I/O.</b> Port 0 is the multiplexed address/data bus. During the time when ALE is high, the LSB of a memory address is presented. When ALE falls, the port transitions to a bidirectional data bus. This bus is used to read external ROM and read/write external RAM memory or peripherals. The Port 0 has no true port latch and can not be written directly by software. The reset condition of Port 0 is high. No pullup resistors are needed.																		
1–8	2–9	40–44 1–3	P1.0–P1.7	<b>Port 1 – I/O.</b> Port 1 functions as both a 8-bit bidirectional I/O port and a alternate functional interface for Timer 2 I/O, new External Interrupts, and new Serial Port 1. The reset condition of Port 1 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS80C320 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port once again becomes the output high (and input) state. The alternate modes of Port 1 are outlined as follows.  <table><tr><th>Port</th><th>Alternate Function</th></tr><tr><td>P1.0</td><td>T2 External I/O for Timer/Counter 2</td></tr><tr><td>P1.1</td><td>T2EX Timer/Counter 2 Capture/Reload Trigger</td></tr><tr><td>P1.2</td><td>RXD1 Serial Port 1 Input</td></tr><tr><td>P1.3</td><td>TXD1 Serial Port 1 Output</td></tr><tr><td>P1.4</td><td>INT2 External Interrupt 2 (Positive Edge Detect)</td></tr><tr><td>P1.5</td><td>INT3 External Interrupt 3 (Negative Edge Detect)</td></tr><tr><td>P1.6</td><td>INT4 External Interrupt 4 (Positive Edge Detect)</td></tr><tr><td>P1.7</td><td>INT5 External Interrupt 5 (Negative Edge Detect)</td></tr></table>	Port	Alternate Function	P1.0	T2 External I/O for Timer/Counter 2	P1.1	T2EX Timer/Counter 2 Capture/Reload Trigger	P1.2	RXD1 Serial Port 1 Input	P1.3	TXD1 Serial Port 1 Output	P1.4	INT2 External Interrupt 2 (Positive Edge Detect)	P1.5	INT3 External Interrupt 3 (Negative Edge Detect)	P1.6	INT4 External Interrupt 4 (Positive Edge Detect)	P1.7	INT5 External Interrupt 5 (Negative Edge Detect)
Port	Alternate Function																					
P1.0	T2 External I/O for Timer/Counter 2																					
P1.1	T2EX Timer/Counter 2 Capture/Reload Trigger																					
P1.2	RXD1 Serial Port 1 Input																					
P1.3	TXD1 Serial Port 1 Output																					
P1.4	INT2 External Interrupt 2 (Positive Edge Detect)																					
P1.5	INT3 External Interrupt 3 (Negative Edge Detect)																					
P1.6	INT4 External Interrupt 4 (Positive Edge Detect)																					
P1.7	INT5 External Interrupt 5 (Negative Edge Detect)																					
1 2 3 4 5 6 7 8	2 3 4 5 6 7 8 9	40 41 42 43 44 1 2 3																				

DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION																		
21	24	18	A8 (P2.0)	<b>A15–A8 (Port 2) – Output.</b> Port 2 serves as the MSB for external addressing. P2.7 is A15 and P2.0 is A8. The DS80C320 will automatically place the MSB of an address on P2 for external ROM and RAM access. Although Port 2 can be accessed like an ordinary I/O port, the value stored on the Port 2 latch will never be seen on the pins (due to memory access). Therefore writing to Port 2, in software is only useful for the instructions MOVX A, @Ri or MOVX @Ri, A. These instructions use the Port 2 internal latch to supply the external address MSB. In this case, the Port 2 latch value will be supplied as the address information.																		
22	25	19	A9 (P2.1)																			
23	26	20	A10 (P2.2)																			
24	27	21	A11 (P2.3)																			
25	28	22	A12 (P2.4)																			
26	29	23	A13 (P2.5)																			
27	30	24	A14 (P2.6)																			
28	31	25	A15 (P2.7)																			
10–17	11, 13–19	5, 7–13	P3.0–P3.7	<b>Port 3 – I/O.</b> Port 3 functions as both a 8-bit bidirectional I/O port and an alternate functional interface for External Interrupts, Serial Port 0, Timer 0 & 1 Inputs, RD and WR strobes. The reset condition of Port 3 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS80C320 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port once again becomes both the output high and input state. The alternate modes of Port 3 are outlined below.  <table><tr><th>Port</th><th>Alternate Mode</th></tr><tr><td>P3.0</td><td>RXD0 Serial Port 0 Input</td></tr><tr><td>P3.1</td><td>TXD0 Serial Port 0 Output</td></tr><tr><td>P3.2</td><td>INT0 External Interrupt 0</td></tr><tr><td>P3.3</td><td>INT1 External Interrupt 1</td></tr><tr><td>P3.4</td><td>T0 Timer 0 External Input</td></tr><tr><td>P3.5</td><td>T1 Timer 1 External Input</td></tr><tr><td>P3.6</td><td>WR External Data Memory Write Strobe</td></tr><tr><td>P3.7</td><td>RD External Data Memory Read Strobe</td></tr></table>	Port	Alternate Mode	P3.0	RXD0 Serial Port 0 Input	P3.1	TXD0 Serial Port 0 Output	P3.2	INT0 External Interrupt 0	P3.3	INT1 External Interrupt 1	P3.4	T0 Timer 0 External Input	P3.5	T1 Timer 1 External Input	P3.6	WR External Data Memory Write Strobe	P3.7	RD External Data Memory Read Strobe
Port	Alternate Mode																					
P3.0	RXD0 Serial Port 0 Input																					
P3.1	TXD0 Serial Port 0 Output																					
P3.2	INT0 External Interrupt 0																					
P3.3	INT1 External Interrupt 1																					
P3.4	T0 Timer 0 External Input																					
P3.5	T1 Timer 1 External Input																					
P3.6	WR External Data Memory Write Strobe																					
P3.7	RD External Data Memory Read Strobe																					
10	11	5																				
11	13	7																				
12	14	8																				
13	15	9																				
14	16	10																				
15	17	11																				
16	18	12																				
17	19	13																				
31	35	29	EA	<b>EA – Input.</b> This pin must be connected to ground for proper operation.																		
–	12, 34	6, 28	NC	<b>NC – Reserved.</b> These pins should not be connected. They are reserved for use with future devices in this family.																		
–	1	39		<b>NC – Reserved.</b> These pins are reserved for additional ground pins on future products.																		

### 80C32 COMPATIBILITY

The DS80C320 is a CMOS 80C32 compatible microcontroller designed for high performance. In most cases the DS80C320 can drop into an existing 80C32 design to significantly improve the operation. Every effort has been made to keep the device familiar to 8032 users, yet it has many new features. In general, software written for existing 80C32 based systems will work on the DS80C320. The exception is critical timing since the High-Speed Microcontroller performs its instructions much faster than the original. It may be necessary to use memories with faster access times if the same crystal frequency is used.

The DS80C320 runs the standard 8051 instruction set and is pin compatible with an 80C32 in any of three standard packages. The DS80C320 also provides the same timer/counter resources, full-duplex serial port, 256 bytes of scratchpad RAM and I/O ports as the standard 80C32. Timers will default to a 12 clock per cycle operation to keep timing compatible with original 8051 systems. However, they can be programmed to run at the new 4 clocks per cycle if desired.

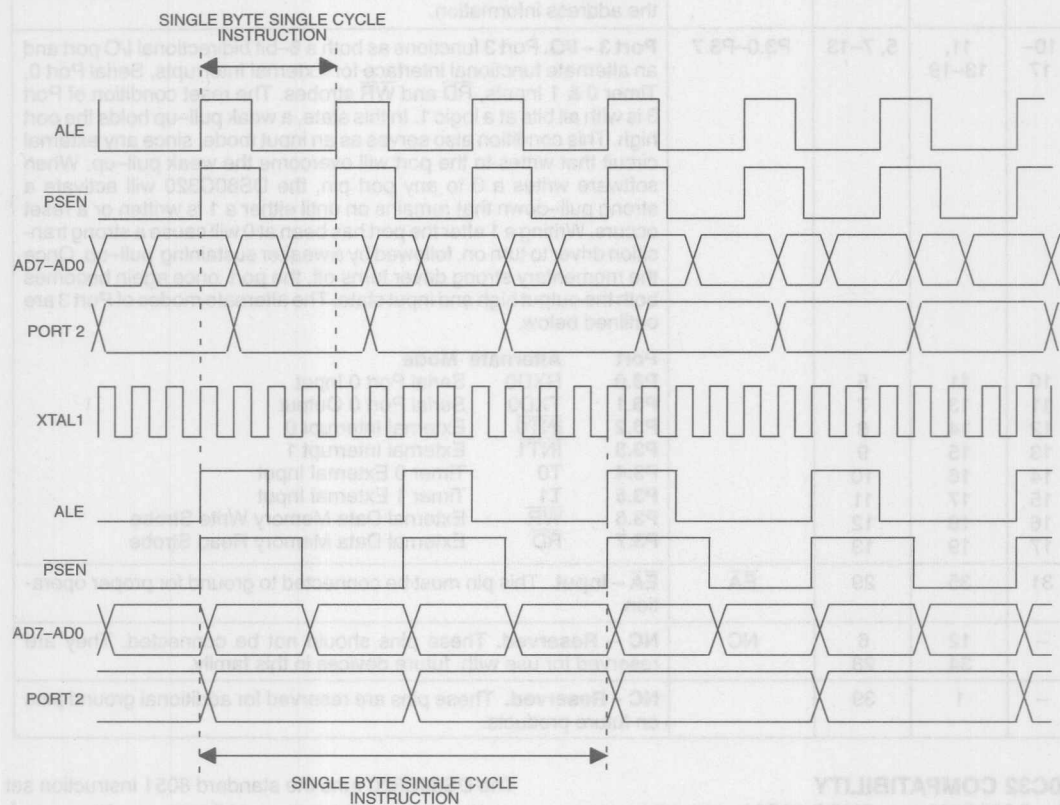
New hardware features are accessed using Special Function Registers that do not overlap with standard 80C32 locations. A summary of these SFRs is provided below.

The DS80C320 addresses memory in an identical fashion to the standard 80C32. Electrical timing will appear different due to the high speed nature of the product. However, the signals are essentially the same. Detailed timing diagrams are provided below in the electrical specifications.

This data sheet assumes the user is familiar with the basic features of the standard 80C32. In addition to these standard features, the DS80C320 includes many new functions. This data sheet provides only a summary and overview. Detailed descriptions are available in the High-Speed Microcontroller User's Guide.

## COMPARATIVE TIMING OF THE DS80C320 AND 80C32 Figure 2

### DS80C320 TIMING



### STANDARD 80C32 TIMING

## HIGH-SPEED OPERATION

The DS80C320 is built around a high speed 80C32 compatible core. Higher speed comes not just from increasing the clock frequency, but from a newer, more efficient design.

In this updated core, dummy memory cycles have been eliminated. In a conventional 80C32, machine cycles are generated by dividing the clock frequency by 12. In the DS80C320, the same machine cycle is performed in 4 clocks. Thus the fastest instruction, 1 machine cycle, is executed three times faster for the same crystal frequency. Note that these are identical instructions. A comparison of the timing differences is shown in Figure 2. The majority of instructions on the DS80C320 will see the full 3 to 1 speed improvement. Some instructions will get between 1.5 and 2.4 X improvement. Note that all instructions are faster than the original 80C51. Table 2 below shows a summary of the instruction set including the speed.

The numerical average of all opcodes is approximately a 2.5 to 1 speed improvement. Individual programs will be affected differently, depending on the actual instructions used. Speed sensitive applications would make the most use of instructions that are three times faster. However, the sheer number of 3 to 1 improved opcodes makes dramatic speed improvements likely for any code. When these architecture improvements are combined with 0.8  $\mu$ m CMOS, the result is a single cycle instruction execution in 160 ns. The Dual Data Pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

## INSTRUCTION SET SUMMARY

All instructions in the DS80C320 perform the same functions as their 80C32 counterparts. Their affect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute and relative number of clocks.

For absolute timing of real-time events, the timing of software loops will need to be calculated using the table

below. However, counter/timers default to run at the older 12 clocks per increment. Therefore, while software runs at higher speed, timer-based events need no modification to operate as before. Timers can be set to run at 4 clocks per increment cycle to take advantage of higher speed operation.

The relative time of two instructions might be different in the new architecture than it was previously. For example, in the original architecture, the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction used two machine cycles or 24 oscillator cycles. Therefore, they required the same amount of time. In the DS80C320, the MOVX instruction can be done in two machine cycles or eight oscillator cycles but the "MOV direct, direct" uses three machine cycles or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times from each other. This is because in most cases, the DS80C320 uses one cycle for each byte. The user concerned with precise program timing should examine the timing of each instruction for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV.

## INSTRUCTION SET SUMMARY Table 2

### Legends:

A	—	Accumulator
Rn	—	Register R7–R0
direct	—	Internal Register address
@Ri	—	Internal Register pointed-to by R0 or R1 (except MOVX)
rel	—	2's complement offset byte
bit	—	direct bit-address
#data	—	8-bit constant
#data 16	—	16-bit constant
addr 16	—	16-bit destination address
addr 11	—	11-bit destination address



INSTRUCTION	BYTE	OSCILLATOR CYCLES
<b>Arithmetic Instructions:</b>		
ADD A, Rn	1	4
ADD A, direct	2	8
ADD A, @Ri	1	4
ADD A, #data	2	8
ADDC A, Rn	1	4
ADDC A, direct	2	8
ADDC A, @Ri	1	4
ADDC A, #data	2	8
SUBB A, Rn	1	4
SUBB A, direct	2	8
SUBB A, @Ri	1	4
SUBB A, #data	2	8

**Logical Instructions:**

ANL A, Rn	1	4
ANL A, direct	2	8
ANL A, @Ri	1	4
ANL A, #data	2	8
ANL direct, A	2	8
ANL direct, #data	3	12
ORL A, Rn	1	4
ORL A, direct	2	8
ORL A, @Ri	1	4
ORL A, #data	2	8
ORL direct, A	2	8
ORL direct, #data	3	12

**Data Transfer****Instructions:**

MOV A, Rn	1	4
MOV A, direct	2	8
MOV A, @Ri	1	4
MOV A, #data	2	8
MOV Rn, A	1	4
MOV Rn, direct	2	8
MOV Rn, #data	2	8
MOV direct, A	2	8
MOV direct, Rn	2	8
MOV direct1, direct2	3	12
MOV direct, @Ri	2	8
MOV direct, #data	3	12
MOV @Ri, A	1	4
MOV @Ri, direct	2	8
MOV @Ri, #data	2	8
MOV DPTR, #data 16	3	12

\*User Selectable

INSTRUCTION	BYTE	OSCILLATOR CYCLES
INC A	1	4
INC Rn	1	4
INC direct	2	8
INC @Ri	1	4
INC DPTR	1	12
DEC A	1	4
DEC Rn	1	4
DEC direct	2	8
DEC @Ri	1	4
MUL AB	1	20
DIV AB	1	20
DA A	1	4
XRL A, Rn	1	4
XRL A, direct	2	8
XRL A, @Ri	1	4
XRL A, #data	2	8
XRL direct, A	2	8
XRL direct, #data	3	12
CLR A	1	4
CPL A	1	4
RL A	1	4
RLC A	1	4
RR A	1	4
RRC A	1	4
SWAP A	1	4
MOVC A, @A+DPTR	1	12
MOVC A, @A+PC	1	12
MOVX A, @Ri	1	8-36 *
MOVX A, @DPTR	1	8-36 *
MOVX @Ri, A	1	8-36 *
MOVX @DPTR, A	1	8-36 *
PUSH direct	2	8
POP direct	2	8
XCH A, Rn	1	4
XCH A, direct	2	8
XCH A, @Ri	1	4
XCHD A, @Ri	1	4

**Bit Manipulation****Instructions:**

CLR C	1	4	ANL C, bit	2	8
CLR bit	2	8	ANL C, $\overline{\text{bit}}$	2	8
SETB C	1	4	ORL C, bit	2	8
SETB bit	2	8	ORL C, $\overline{\text{bit}}$	2	8
CPL C	1	4	MOV C, bit	2	8
CPL bit	2	8	MOV bit, C	2	8

**Program Branching****Instructions:**

ACALL addr 11	2	12	CJNE A, direct, rel	3	16
LCALL addr 16	3	16	CJNE A, #data, rel	3	16
RET	1	16	CJNE Rn, #data, rel	3	16
RETI	1	16	CJNE Ri, #data, rel	3	16
AJMP addr 11	2	12	NOP	1	4
LJMP addr 16	3	16	JC rel	2	12
SJMP rel	2	12	JNC rel	2	12
JMP @A+DPTR	1	12	JB bit, rel	3	16
JZ rel	2	12	JNB bit, rel	3	16
JNZ rel	2	12	JBC bit, rel	3	16
DJNZ Rn, rel	2	12			
DJNZ direct, rel	3	16			

The table above shows the speed for each class of instruction. Note that many of the instructions have multiple opcodes. There are 255 opcodes for 111 instructions. Of the 255 opcodes, 159 are three times faster than the original 80C32. While a system that emphasizes those instructions will see the most improvement, the large total number that receive a 3 to 1 improvement assure a dramatic speed increase for any system. The speed improvement summary is provided below.

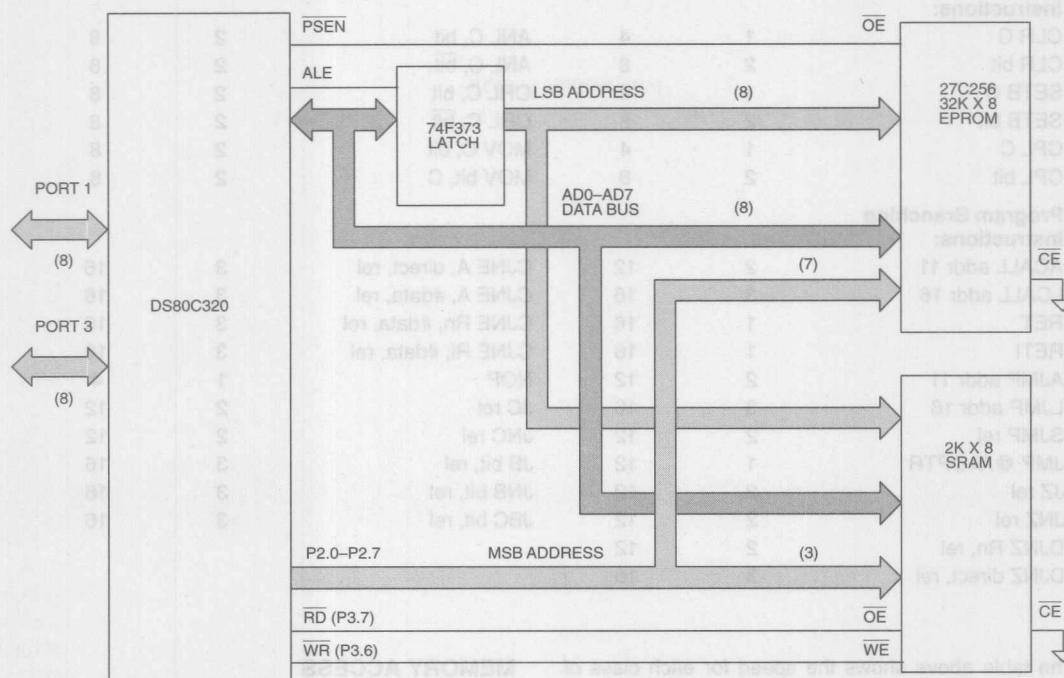
**SPEED ADVANTAGE SUMMARY**

#Opcodes	Speed Improvement
159	3.0 x
51	1.5 x
43	2.0 x
2	2.4 x
255	Average: 2.5

**MEMORY ACCESS**

The DS80C320 contains no on-chip ROM and 256 bytes of scratchpad RAM. Off-chip memory is accessed using the multiplexed address/data bus on P0 and the MSB address on P2. A typical memory connection is shown in Figure 3. Timing diagrams are provided in the Electrical Specifications. Program memory (ROM) is accessed at a fixed rate determined by the crystal frequency and the actual instructions. As mentioned above, an instruction cycle requires four clocks. Data memory (RAM) is accessed according to a variable speed MOVX instruction as described below.

TYPICAL MEMORY CONNECTION Figure 3



### STRETCH MEMORY CYCLE

The DS80C320 allows the application software to adjust the speed of data memory access. The micro is capable of performing the MOVX in as little as two instruction cycles. However, this value can be stretched as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic. Even in high-speed systems, it may not be necessary or desirable to perform data memory access at full speed. In addition, there are a variety of memory mapped peripherals such as LCD displays or UARTs that are not fast.

The Stretch MOVX is controlled by the Clock Control Register at SFR location 8Eh as described below. This allows the user to select a stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. A Stretch of seven will result in a MOVX of nine machine cycles. Software can dynamically change this value depending on the particular memory or peripheral.

On reset, the Stretch value will default to a one resulting in a three cycle MOVX. Therefore, RAM access will not be performed at full speed. This is a convenience to

existing designs that may not have fast RAM in place. When maximum speed is desired, the software should select a Stretch value of zero. When using very slow RAM or peripherals, a larger stretch value can be selected. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven causes the microcontroller to stretch the read/write strobe and all related timing. This results in a wider read/write strobe allowing more time for memory/peripherals to respond. The timing of the variable speed MOVX is shown in the Electrical Specifications. Note that full speed access is not the reset default case. Table 3 below shows the resulting strobe widths for each Stretch value. The memory stretch is implemented using the Clock Control Special Function Register at SFR location 8Eh. The stretch value is selected using bits CKCON.2-0. In the table, these bits are referred to as M2 through M0. The first stretch (default) allows the use of common 120 ns or 150 ns RAMs without dramatically lengthening the memory access.

DATA MEMORY CYCLE STRETCH VALUES Table 3

CKCON.2-0			MEMORY CYCLES	RD or WR STROBE WIDTH IN CLOCKS	STROBE WIDTH TIME @ 25 MHz
MD2	MD1	MD0			
0	0	0	2	2	80 ns
0	0	1	3 (default)	4	160 ns
0	1	0	4	8	320 ns
0	1	1	5	12	480 ns
1	0	0	6	16	640 ns
1	0	1	7	20	800 ns
1	1	0	8	24	960 ns
1	1	1	9	28	1120 ns

### DUAL DATA POINTER

Data memory block moves can be accelerated using the DS80C320 Dual Data Pointer (DPTR). The standard 8032 DPTR is a 16-bit value that is used to address off-chip data RAM or peripherals. In the DS80C320, the standard data pointer is called DPTR 0 and is located at SFR addresses 82h and 83h. These are the standard locations. No modification of standard code is needed to use DPTR. The new DPTR is located at SFR 84h and 85h and is called DPTR1. The DPTR Select bit (DPS) chooses the active pointer and is located at the LSB of the SFR location 86h. No other bits in register 86h have any effect and are set to 0. The user switches between data pointers by toggling the LSB of register 86h. The increment (INC) instruction is the fastest way to accomplish this. All DPTR-related instructions use the currently selected DPTR for any activity. Therefore only one instruction is required to switch from a source to a destination address. Using the Dual-Data Pointer saves code from needing to save source and destination addresses when doing a block move. Once

loaded, the software simply switches between DPTR and 1. The relevant register locations are as follows.

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (LSB)

Sample code listed below illustrates the saving from using the dual DPTR. The example program was original code written for an 8051 and requires a total of 1869 machine cycles on the DS80C320. This takes 299  $\mu$ s to execute at 25 MHz. The new code using the Dual DPTR requires only 1097 machine cycles taking 175.5  $\mu$ s. The Dual DPTR saves 772 machine cycles or 123.5  $\mu$ s for a 64 byte block move. Since each pass through the loop saves 12 machine cycles when compared to the single DPTR approach, larger blocks gain more efficiency using this feature.

### 64 BYTE BLOCK MOVE WITHOUT DUAL DATA POINTER

; SH and SL are high and low byte source address.

; DH and DL are high and low byte of destination address.

			# CYCLES
MOV	R5, #64d	; NUMBER OF BYTES TO MOVE	2
MOV	DPTR, #SHSL	; LOAD SOURCE ADDRESS	3
MOV	R1, #SL	; SAVE LOW BYTE OF SOURCE	2
MOV	R2, #SH	; SAVE HIGH BYTE OF SOURCE	2
MOV	R3, #DL	; SAVE LOW BYTE OF DESTINATION	2
MOV	R4, #DH	; SAVE HIGH BYTE OF DESTINATION	2
MOVE:			
; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64			
MOVX	A, @DPTR	; READ SOURCE DATA BYTE	2
MOV	R1, DPL	; SAVE NEW SOURCE POINTER	2
MOV	R2, DPH		2

MOV	DPL, R3	; LOAD NEW DESTINATION	2
MOV	DPH, R4		2
MOVX	@DPTR, A	; WRITE DATA TO DESTINATION	2
INC	DPTR	; NEXT DESTINATION ADDRESS	3
MOV	R3, DPL	; SAVE NEW DESTINATION POINTER	2
MOV	R4, DPH		2
MOV	DPL, R1	; GET NEW SOURCE POINTER	2
MOV	DPH, R2		2
INC	DPTR	; NEXT SOURCE ADDRESS	3
DJNZ	R5, MOVE	; FINISHED WITH TABLE?	3

### 64 BYTE BLOCK MOVE WITH DUAL DATA POINTER

; SH and SL are high and low byte source address.  
 ; DH and DL are high and low byte of destination address.  
 ; DPS is the data pointer select. Reset condition is DPS=0, DPTR0 is selected.  
 # CYCLES

EQU	DPS, #86h	; TELL ASSEMBLER ABOUT DPS	
MOV	R5, #64	; NUMBER OF BYTES TO MOVE	2
MOV	DPTR, #DHDH	; LOAD DESTINATION ADDRESS	3
INC	DPS	; CHANGE ACTIVE DPTR	2
MOV	DPTR, #SHSL	; LOAD SOURCE ADDRESS	2

MOVE:

; THIS LOOP IS PERFORMED THE NUMBER OF TIMES LOADED INTO R5, IN THIS EXAMPLE 64

MOVX	A, @DPTR	; READ SOURCE DATA BYTE	2
INC	DPS	; CHANGE DPTR TO DESTINATION	2
MOVX	@DPTR, A	; WRITE DATA TO DESTINATION	2
INC	DPTR	; NEXT DESTINATION ADDRESS	3
INC	DPS	; CHANGE DATA POINTER TO SOURCE	2
INC	DPTR	; NEXT SOURCE ADDRESS	3
DJNZ	R5, MOVE	; FINISHED WITH TABLE?	3

### PERIPHERAL OVERVIEW

Peripherals in the DS80C320 are accessed using Special Function Registers (SFRs). The DS80C320 provides several of the most commonly needed peripheral functions in microcomputer-based systems. These functions are new to the 80C32 family and include a second serial port, Power-fail Reset, Power-fail Interrupt, and a programmable Watchdog Timer. These are described below, and more details are available in the High-Speed Microcontroller User's Guide.

### SERIAL PORTS

The DS80C320 provides a serial port (UART) that is identical to the 80C32. Many applications require serial communication with multiple devices. Therefore the DS80C320 provides a second hardware serial port that is a full duplicate of the standard one. It optionally uses pins P1.2 (RXD1) and P1.3 (TXD1). This port has duplicate control functions included in new SFR locations.

The second serial port operates in a comparable manner with the first. Both can operate simultaneously but can be at different baud rates.

The second serial port has similar control registers (SCON1 at C0h, SBUF1 at C1h) to the original. One difference is that for timer based baud rates, the original serial port can use Timer 1 or Timer 2 to generate baud rates. This is selected via SFR bits. The new serial port can only use Timer 1.

### TIMER RATE CONTROL

One important difference exists between the DS80C320 and 80C32 regarding timers. The original 80C32 used a 12 clock per cycle scheme for timers and consequently for some serial baud rates (depending on the mode). The DS80C320 architecture normally runs using 4 clocks per cycle. However, in the area of timers, the DS80C320 will default to a 12 clock per cycle



scheme on a reset. This allows existing code with real-time dependencies such as baud rates to operate properly. If an application needs higher speed timers or serial baud rates, the timers can be set to run at the 4 clock rate.

The Clock Control register (CKCON – 8Eh) determines these timer speeds. When the relevant CKCON bit is a logic 1, the DS80C320 uses 4 clocks per cycle to generate timer speeds. When the control bit is set to a 0, the DS80C320 uses 12 clocks for timer speeds. The reset condition is a 0. CKCON.5 selects the speed of Timer 2. CKCON.4 selects Timer 1 and CKCON.3 selects Timer 0. Note that unless a user desires very fast timing, it is unnecessary to alter these bits. Note that the timer controls are independent.

### POWER FAIL RESET

The DS80C320 incorporates a precision band-gap voltage reference to determine when  $V_{CC}$  is out-of-tolerance. While powering up, internal circuits will hold the DS80C320 in a reset state until  $V_{CC}$  rises above the  $V_{RST}$  reset threshold. Once  $V_{CC}$  is above this level, the oscillator will begin running. An internal reset circuit will then count 65536 clocks to allow time for power and the oscillator to stabilize. The microcontroller will then exit the reset condition. No external components are needed to generate a power on reset. During power down or during a severe power glitch, as  $V_{CC}$  falls below  $V_{RST}$ , the microcontroller will also generate its own reset. It will hold the reset condition as long as power remains below the threshold. This reset will occur automatically, needing no action from the user or from the software. Refer to the Electrical Specifications for the exact value of  $V_{RST}$ .

### POWER FAIL INTERRUPT

The same reference that generates a precision reset threshold can also generate an optional early warning

Power-fail Interrupt (PFI). When enabled by the application software, this interrupt always has the highest priority. On detecting that the  $V_{CC}$  has dropped below  $V_{PFW}$  and that the PFI is enabled, the processor will vector to ROM address 0033h. The PFI enable is located in the Watchdog Control SFR (WDCON – D8h). Setting WDCON.5 to a logic one will enable the PFI. The application software can also read a flag at WDCON.4. This bit is set when a PFI condition has occurred. The flag is independent of the interrupt enable and software must manually clear it.

### WATCHDOG TIMER

For applications that can not afford to run out-of-control, the DS80C320 incorporates a programmable Watchdog Timer circuit. It resets the uC if software fails to reset the Watchdog before the selected time interval has elapsed. The user selects one of four time-out values. After enabling the Watchdog, software must reset the timer prior to expiration of the interval, or the CPU will be reset. Both the Watchdog Enable and the Watchdog Reset bits are protected by a "Timed Access" circuit. This prevents accidentally clearing the Watchdog. Time-out values are precise since they are related to the crystal frequency as shown below in Table 4. For reference, the time periods at 25 MHz are also shown.

The DS80C320 Watchdog also provides a useful option for systems that may not require a reset. If enabled, then 512 clocks before giving a reset, the Watchdog will give an interrupt. The interrupt can also serve as a convenient time-base generator, or be used to wake-up the processor from Idle mode. The Watchdog function is controlled in the Clock Control (CKCON – 8Eh), Watchdog Control (WDCON – D8h), and Extended Interrupt Enable (EIE – E8h) SFRs. CKCON.7 and CKCON.6 are called WD1 and WD0 respectively and are used to select the Watchdog time-out period as shown in Table 4.

**WATCHDOG TIME-OUT VALUES** Table 4

WD1	WD0	INTERRUPT TIME-OUT	TIME (@25 MHz)	RESET TIME-OUT	TIME (@25 MHz)
0	0	$2^{17}$ clocks	5.243 ms	$2^{17} + 512$ clocks	5.263 ms
0	1	$2^{20}$ clocks	41.94 ms	$2^{20} + 512$ clocks	41.96 ms
1	0	$2^{23}$ clocks	335.54 ms	$2^{23} + 512$ clocks	335.56 ms
1	1	$2^{26}$ clocks	2684.35 ms	$2^{26} + 512$ clocks	2684.38 ms

As shown above, the Watchdog Timer uses the crystal frequency as a time base. A user selects one of four counter values to determine the time-out. These clock counter lengths are  $2^{17} = 131,072$  clocks;  $2^{20} = 1,048,576$ ;  $2^{23} = 8,388,608$  clocks; or  $2^{26} = 67,108,864$  clocks. The times shown in Table 4 above are with a 25 MHz crystal frequency. Note that once the counter chain has reached a conclusion, the optional interrupt is generated. Regardless of whether the user enables this interrupt, there are then 512 clocks left until a reset occurs. There are five control bits in special function registers that affect the Watchdog Timer and two status flags that report to the user.

WDIF (WDCON.3) is the interrupt flag that is set when there are 512 clocks remaining until a reset occurs. WTRF (WDCON.2) is the flag that is set when a Watchdog reset has occurred. This allows the application software to determine the source of a reset.

EWT (WDCON.1) is the enable for the Watchdog Timer. Software sets this bit to enable the timer. The bit is pro-

tected by Timed Access discussed below. RWT (WDCON.0) is the bit that software uses to restart the Watchdog Timer. Setting this bit restarts the timer for another full interval. Application software must set this bit prior to the time-out. As mentioned previously, WD1 and 0 (CKCON.7 and 6) select the time-out. Finally, the Watchdog Interrupt is enabled using EWDI (EIE.4). The Special Function Register map is shown below.

## INTERRUPTS

The DS80C320 provides 13 sources of interrupt with three priority levels. The Power-fail Interrupt (PFI), if enabled, always has the highest priority. There are two remaining user selectable priorities: high and low. If two interrupts that have the same priority occur simultaneously, the natural precedence given below determines which is acted upon. Except for the PFI, all interrupts that are new to the 8051 family have a lower natural priority than the originals.

**INTERRUPT PRIORITY Table 5**

NAME	DESCRIPTION	VECTOR	NATURAL PRIORITY	OLD/NEW
PFI	Power Fail Interrupt	33h	1	NEW
INT0	External Interrupt 0	03h	2	OLD
TF0	Timer 0	0Bh	3	OLD
INT1	External Interrupt 1	13h	4	OLD
TF1	Timer 1	1Bh	5	OLD
SCON0	T10 or RI0 from serial port 0	23h	6	OLD
TF2	Timer 2	2Bh	7	OLD
SCON1	T11 or RI1 from serial port 1	3Bh	8	NEW
INT2	External Interrupt 2	43h	9	NEW
INT3	External Interrupt 3	4Bh	10	NEW
INT4	External Interrupt 4	53h	11	NEW
INT5	External Interrupt 5	5Bh	12	NEW
WDTI	Watchdog Time-out Interrupt	63h	13	NEW

## POWER MANAGEMENT

The DS80C320 provides the standard Idle and power down (Stop) that are available on the standard 80C32. However the DS80C320 has enhancements that make these modes more useful, and allow more power saving.

The Idle mode is invoked by setting the LSB of the Power Control register (PCON – 87h). Idle will leave internal clocks, serial port and timer running. No memory access will be performed so power is dramati-

cally reduced. Since clocks are running, the Idle power consumption is related to crystal frequency. It should be approximately 1/2 of the operational power. The CPU can exit the Idle state with any interrupt or a reset.

The power-down or Stop mode is invoked by setting the PCON.1 bit. Stop mode is a lower power state than Idle since it turns off all internal clocking. The  $I_{CC}$  of a standard Stop mode is approximately 1  $\mu A$  but is specified in the Electrical Specifications. The CPU will exit Stop mode from an external interrupt or a reset condition.

Note that internally generated interrupts (timer, serial port, watchdog) are not useful since they require clocking activity.

### IDLE MODE ENHANCEMENTS

A simple enhancement to Idle mode makes it substantially more useful. The innovation involves not the Idle mode itself, but the watchdog timer. As mentioned above, the Watchdog Timer provides an optional interrupt capability. This interrupt can provide a periodic interval timer to bring the DS80C320 out of Idle mode. This can be useful even if the Watchdog is not normally used. By enabling the Watchdog Timer and its interrupt prior to invoking Idle, a user can periodically come out of Idle perform an operation, then return to Idle until the next operation. This will lower the overall power consumption. When using the Watchdog Interrupt to cancel the Idle state, make sure to restart the Watchdog Timer or it will cause a reset.

### STOP MODE ENHANCEMENTS

The DS80C320 provides two enhancements to the Stop mode. As documented above, the DS80C320 provides a band-gap reference to determine Power-fail Interrupt and Reset thresholds. The default state is that the band-gap reference is off when Stop mode is invoked. This allows the extremely low power state mentioned above. A user can optionally choose to have the band-gap enabled during Stop mode. This means that PFI and power-fail reset will be activated and are valid means for leaving Stop mode.

In Stop mode with the band-gap on,  $I_{CC}$  will be approximately 50  $\mu A$  compared with 1  $\mu A$  with the band-gap off. If a user does not require a Power-fail Reset or Interrupt while in Stop mode, the band-gap can remain turned off. Note that only the most power sensitive applications should turn off the band-gap, as this results in an uncontrolled power down condition.

The control of the band-gap reference is located in the Extended Interrupt Flag register (EXIF – 91h). Setting BGS (EXIF.0) to a one will leave the band-gap reference enabled during Stop mode. The default or reset condition is with the bit at a logic 0. This results in the band-gap being turned off during Stop mode. Note that

this bit has no control of the reference during full power or Idle modes.

The second feature allows an additional power saving option. This is the ability to start instantly when exiting Stop mode. It is accomplished using an internal ring oscillator that can be used when exiting Stop mode in response to an interrupt. The benefit of the ring oscillator is as follows.

Using Stop mode turns off the crystal oscillator and all internal clocks to save power. This requires that the oscillator be restarted when exiting Stop mode. Actual start-up time is crystal dependent, but is normally at least 4 ms. A common recommendation is 10 ms. In an application that will wake-up, perform a short operation, then return to sleep, the crystal start-up can be longer than the real transaction. However, the ring oscillator will start instantly. The user can perform a simple operation and return to sleep before the crystal has even stabilized. If the ring is used to start and the processor remains running, hardware will automatically switch to the crystal once a power-on reset interval (65536 clocks) has expired. This value is used to guarantee stability even though power is not being cycled.

If the user returns to Stop mode prior to switching of crystal, then all clocks will be turned off again. The ring oscillator runs at approximately 4 MHz but will not be a precision value. No real-time precision operations (including serial communication) should be conducted during this ring period. Figure 7 shows how the operation would compare when using the ring, and when starting up normally. The default state is to come out of Stop mode without using the ring oscillator.

This function is controlled using the RGSL – Ring Select bit at EXIF.1 (EXIF – 91h). When EXIF.1 is set, the ring oscillator will be used to come out of Stop mode quickly. As mentioned above, the processor will automatically switch from the ring (if enabled) to the crystal after a delay of 65536 crystal clocks. For a 3.57 MHz crystal, this is approximately 18 ms. The processor sets a flag called RGMD – Ring Mode to tell software that the ring is being used. This bit at EXIF.2 will be a logic 1 when the ring is in use. No serial communication or precision timing should be attempted while this bit is set, since the operating frequency is not precise.

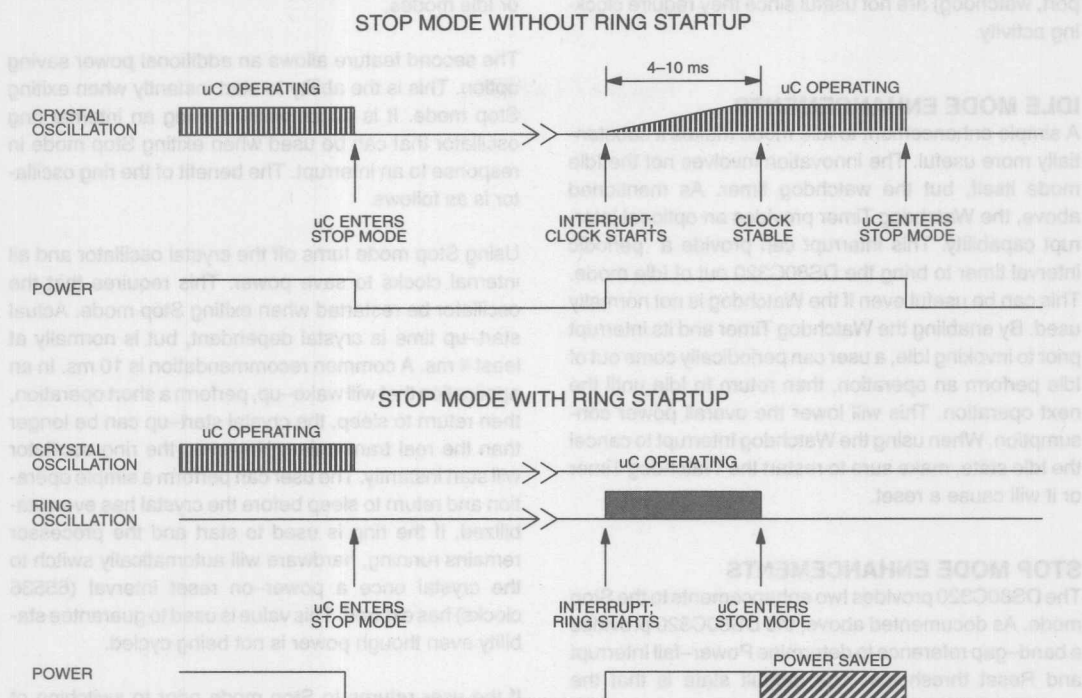
**RING OSCILLATOR START-UP Figure 4**

Diagram assumes that the operation following Stop requires less than 18 ms complete.

### TIMED ACCESS PROTECTION

Selected SFR bits are critical to operation, making it desirable to protect against an accidental write operation. The Timed Access procedure prevents an errant cpu from accidentally altering a bit that would cause difficulty. The Timed Access procedure requires that the write of a protected bit be preceded by the following instructions :

```
MOV    0C7h, #0AAh
MOV    0C7h, #55h
```

By writing an AAh followed by a 55h to the Timed Access register (location C7h), the hardware opens a two cycle window that allows software to modify one of the protected bits. If the instruction that seeks to modify the protected bit is not immediately proceeded by these instructions, the write will not take effect. The protected bits are:

EXIF.0	BGS Band-gap Select
WDCON.6	POR Power-on Reset flag
WDCON.1	EWT Enable Watchdog
WDCON.0	RWT Reset Watchdog
WDCON.3	WDIF Watchdog Interrupt Flag

### SPECIAL FUNCTION REGISTERS

Most special features of the DS80C320 or 80C32 are controlled by bits in special function registers (SFRs). This allows the DS80C320 to add many features but use the same instruction set. When writing software to use a new feature, the SFR must be defined to an assembler or compiler using an equate statement. This is the only change needed to access the new function. The DS80C320 duplicates the SFRs that are contained in the standard 80C32. Table 6 shows the register addresses and bit locations. Many are standard 80C32 registers. The High-Speed Microcontroller User's Guide describes all SFRs.



SPECIAL FUNCTION REGISTER LOCATIONS Table 6

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SP									81h
DPL									82h
DPH									83h
DPL1									84h
DPH1									85h
DPS	0	0	0	0	0	0	0	SEL	86h
PCON	SMOD_0	SMOD0	–	–	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
CKCON	WD1	WD0	T2M	T1M	T0M	MD2	MD1	MD0	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
EXIF	IE5	IE4	IE3	IE2	–	RGMD	RGSL	BGS	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.0	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	C0h
SBUF1									C1h
STATUS	PIP	HIP	LIP	1	1	1	1	1	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	–	–	–	–	–	–	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
ACC									E0h
EIE	–	–	–	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	–	–	–	PWDI	PX5	PX4	PX3	PX2	F8h



**ELECTRICAL SPECIFICATIONS** $V_{CC}=+5V \pm 10\%$ ;  $t_A=0^{\circ}C$  to  $70^{\circ}C$ **ABSOLUTE MAXIMUM RATINGS\***

Voltage on Any Pin Relative to Ground

-1.0V to 7.0V

Operating Temperature

-40°C to +85°C

Storage Temperature

-55°C to 125°C

Soldering Temperature

260°C for 10 seconds

\* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

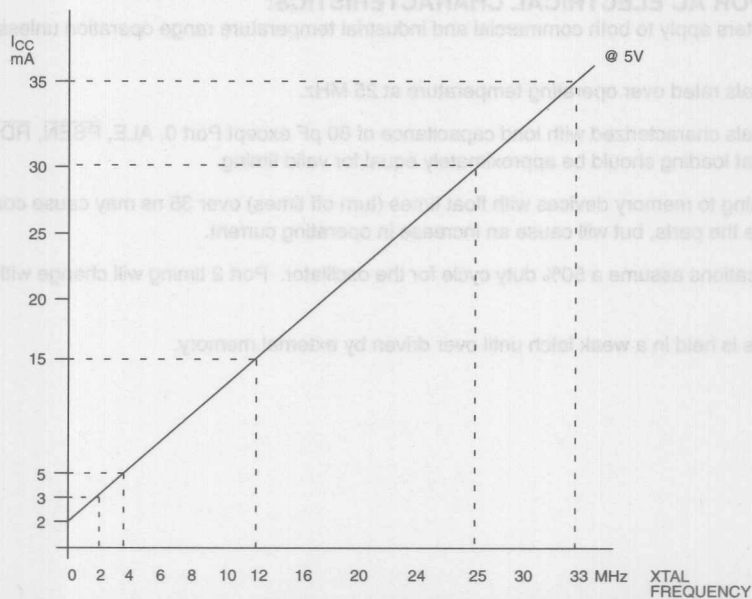
**DC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to  $5.5V$ )

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Operating Supply Voltage	$V_{CC}$	4.5	5.0	5.5	V	1
Power Fail Warning	$V_{PFW}$	4.25	4.38	4.5	V	1
Minimum Operating Voltage	$V_{RST}$	4.0	4.1	4.25	V	1
Supply Current Active Mode @ 25 MHz	$I_{CC}$		30	45	mA	2
Supply Current Idle Mode @ 25 MHz	$I_{IDLE}$		15	25	mA	3
Supply Current Active Mode @ 33 MHz	$I_{CC}$		35		mA	2
Supply Current Idle Mode @ 33 MHz	$I_{IDLE}$		20		mA	3
Supply Current Stop Mode, Band-gap Reference Disabled	$I_{STOP}$		.01	1	$\mu A$	4
Supply Current Stop Mode, Band-gap Reference Enabled	$I_{SPBG}$		50	80	$\mu A$	4, 10
Input Low Level	$V_{IL}$	-0.3		+0.8	V	1
Input High Level (Except XTAL1 and RST)	$V_{IH1}$	2.0		$V_{CC}+0.3$	V	1
Input High Level XTAL1 and RST	$V_{IH2}$	3.5		$V_{CC}+0.3$	V	1
Output Low Voltage Ports 1, 3 @ $I_{OL}=1.6$ mA	$V_{OL1}$			0.45	V	1
Output Low Voltage Ports 0, 2, ALE, PSEN @ $I_{OL}=3.2$ mA	$V_{OL2}$			0.45	V	1, 5
Output High Voltage Ports 0, 1, 2, 3, ALE, PSEN, @ $I_{OH}=50$ $\mu A$	$V_{OH1}$	2.4			V	1, 6
Output High Voltage Ports 1, 3 @ $I_{OH}=1.5$ mA	$V_{OH2}$	2.4			V	1, 7
Output High Voltage Ports 0, 2, ALE, PSEN @ $I_{OH}=8$ mA	$V_{OH3}$	2.4			V	1, 5
Input Low Current Ports 1, 3 @ 0.45V	$I_{IL}$			-55	$\mu A$	
Transition Current from 1 to 0 Ports 1, 3, @ 2V	$I_{TL}$			-650	$\mu A$	8
Input Leakage Port 0, Bus Mode	$I_L$	-300		+300	$\mu A$	9
RST Pull-down Resistance	$R_{RST}$	50		170	k $\Omega$	

**NOTES FOR DC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature operation unless otherwise noted.

1. All voltages are referenced to ground.
2. Active current is measured with a 25 MHz clock source driving XTAL1,  $V_{CC}=RST=5.5V$ , all other pins disconnected.
3. Idle mode current is measured with a 25 MHz clock source driving XTAL1,  $V_{CC}=5.5V$ , RST at ground, all other pins disconnected.
4. Stop mode current measured with XTAL1 and RST grounded,  $V_{CC}=5.5V$ , all other pins disconnected.
5. When addressing external memory.
6. RST=5.5V. This condition mimics operation of pins in I/O mode.
7. During a 0 to 1 transition, a one-shot drives the ports hard for two clock cycles. This measurement reflects port in transition mode.
8. Ports 1, 2, and 3 source transition current when being pulled down externally. It reaches its maximum at approximately 2V.
9.  $0.45 < V_{IN} < V_{CC}$ . Not a high impedance input. This port is a weak address holding latch because Port 0 is dedicated as an address bus on the DS80C320. Peak current occurs near the input transition point of the latch, approximately 2V.
10. Over the industrial temperature range, this specification has a maximum value of 200  $\mu A$ .

**TYPICAL  $I_{CC}$  VERSUS FREQUENCY** Figure 5

**AC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	25 MHz MIN	25 MHz MAX	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNITS
Oscillator Frequency	$1/t_{CLCL}$	0	25	0	25	MHz
ALE Pulse Width	$t_{LHLL}$	50		$1.5t_{CLCL}-10$		ns
Port 0 Address Valid to ALE Low	$t_{AVLL}$	9		$0.5t_{CLCL}-11$		ns
Address Hold After ALE Low	$t_{LLAX1}$	5	note 5	$0.25t_{CLCL}-5$	note 5	ns
Address Hold After ALE Low for MOVX WR	$t_{LLAX2}$	13		$0.5t_{CLCL}-7$		ns
ALE Low to Valid Instruction In	$t_{LLIV}$		73		$2.5t_{CLCL}-27$	ns
ALE Low to $\overline{PSEN}$ Low	$t_{LLPL}$	3		$0.25t_{CLCL}-7$		ns
$\overline{PSEN}$ Pulse Width	$t_{PLPH}$	83		$2.25t_{CLCL}-7$		ns
$\overline{PSEN}$ Low to Valid Instr. In	$t_{PLIV}$		69		$2.25t_{CLCL}-21$	ns
Input Instruction Hold After $\overline{PSEN}$	$t_{PXIX}$	0		0		ns
Input Instruction Float After $\overline{PSEN}$	$t_{PXIZ}$		35		$t_{CLCL}-5$	ns
Port 0 Address to Valid Instr. In	$t_{AVIV1}$		93		$3t_{CLCL}-27$	ns
Port 2 Address to Valid Instr. In	$t_{AVIV2}$		107		$3.5t_{CLCL}-33$	ns
$\overline{PSEN}$ Low to Address Float	$t_{PLAZ}$		note 5		note 5	ns

**NOTES FOR AC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature range operation unless otherwise noted.

1. All signals rated over operating temperature at 25 MHz.
2. All signals characterized with load capacitance of 80 pF except Port 0, ALE,  $\overline{PSEN}$ ,  $\overline{RD}$  and  $\overline{WR}$  at 100 pF. Note that loading should be approximately equal for valid timing.
3. Interfacing to memory devices with float times (turn off times) over 35 ns may cause contention. This will not damage the parts, but will cause an increase in operating current.
4. Specifications assume a 50% duty cycle for the oscillator. Port 2 timing will change with the duty cycle variations.
5. Address is held in a weak latch until over driven by external memory.

## MOVX CHARACTERISTICS

(0°C to 70°C;  $V_{CC}=4.0$  to 5.5V)

PARAMETER	SYMBOL	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNITS	STRETCH
$\overline{RD}$ Pulse Width	$t_{RLRH}$	$2t_{CLCL}-11$ $t_{MCS}-11$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{WR}$ Pulse Width	$t_{WLWH}$	$2t_{CLCL}-11$ $t_{MCS}-11$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Valid Data In	$t_{RLDV}$		$2t_{CLCL}-25$ $t_{MCS}-25$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold After Read	$t_{RHDx}$	0		ns	
Data Float After Read	$t_{RHDZ}$		$t_{CLCL}-5$ $2t_{CLCL}-5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to Valid Data In	$t_{LLDV}$		$2.5t_{CLCL}-26$ $1.5t_{CLCL}-28+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to Valid Data In	$t_{AVDV1}$		$3t_{CLCL}-24$ $2t_{CLCL}-31+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to Valid Data In	$t_{AVDV2}$		$3.5t_{CLCL}-32$ $2.5t_{CLCL}-34+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$t_{LLWL}$	$0.5t_{CLCL}-5$ $1.5t_{CLCL}-5$	$0.5t_{CLCL}+6$ $1.5t_{CLCL}+8$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL1}$	$t_{CLCL}-9$ $2t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL2}$	$1.5t_{CLCL}-9$ $2.5t_{CLCL}-13$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Valid to $\overline{WR}$ Transition	$t_{QVWX}$	-9 $t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold After Write	$t_{WHQX}$	$t_{CLCL}-7$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Address Float	$t_{RLAZ}$		note 5	ns	
$\overline{RD}$ or $\overline{WR}$ High to ALE High	$t_{WHLH}$	0 $t_{CLCL}-5$	10 $t_{CLCL}+11$	ns	$t_{MCS}=0$ $t_{MCS}>0$

NOTE:  $t_{MCS}$  is a time period related to the Stretch memory cycle selection. The following table shows the value of  $t_{MCS}$  for each Stretch selection.

M2	M1	M0	MOVX CYCLES	$t_{MCS}$
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	4 $t_{CLCL}$
0	1	0	4 machine cycles	8 $t_{CLCL}$
0	1	1	5 machine cycles	12 $t_{CLCL}$
1	0	0	6 machine cycles	16 $t_{CLCL}$
1	0	1	7 machine cycles	20 $t_{CLCL}$
1	1	0	8 machine cycles	24 $t_{CLCL}$
1	1	1	9 machine cycles	28 $t_{CLCL}$

**AC ELECTRICAL CHARACTERISTICS UP TO 33 MHz**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	33 MHz MIN	33 MHz MAX	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNITS
Oscillator Frequency	$1/t_{CLCL}$	0	33	0	33	MHz
ALE Pulse Width	$t_{LHLL}$	50		$1.5t_{CLCL}-10$		ns
Port 0 Address Valid to ALE Low	$t_{AVLL}$	9		$.5t_{CLCL}-11$		ns
Address Hold After ALE Low	$t_{LLAX1}$	5	note 5	$.25t_{CLCL}-5$	note 5	ns
Address Hold After ALE Low for MOVX WR	$t_{LLAX2}$	13		$.5t_{CLCL}-7$		ns
ALE Low to Valid Instruction In	$t_{LLIV}$		73		$2.5t_{CLCL}-27$	ns
ALE Low to $\overline{PSEN}$ Low	$t_{LLPL}$	3		$.25t_{CLCL}-7$		ns
$\overline{PSEN}$ Pulse Width	$t_{PLPH}$	83		$2.25t_{CLCL}-7$		ns
$\overline{PSEN}$ Low to Valid Instr. In	$t_{PLIV}$		69		$2.25t_{CLCL}-21$	ns
Input Instruction Hold After $\overline{PSEN}$	$t_{PXIX}$	0		0		ns
Input Instruction Float After $\overline{PSEN}$	$t_{PXIZ}$		35		$t_{CLCL}-5$	ns
Port 0 Address to Valid Instr. In	$t_{AVIV1}$		93		$3t_{CLCL}-27$	ns
Port 2 Address to Valid Instr. In	$t_{AVIV2}$		107		$3.5t_{CLCL}-33$	ns
$\overline{PSEN}$ Low to Address Float	$t_{PLAZ}$		note 5		note 5	ns

**NOTES FOR AC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature range operation unless otherwise noted.

1. All signals rated over operating temperature at 33 MHz.
2. All signals characterized with load capacitance of 80 pF except Port 0, ALE,  $\overline{PSEN}$ ,  $\overline{RD}$  and  $\overline{WR}$  at 100 pF. Note that loading should be approximately equal for valid timing.
3. Interfacing to memory devices with float times (turn off times) over 35 ns may cause contention. This will not damage the parts, but will cause an increase in operating current.
4. Specifications assume a 50% duty cycle for the oscillator. Port 2 timing will change with the duty cycle variations.
5. Address is held in a weak latch until over driven by external memory.

	M0	M1	M2
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1



## MOVX CHARACTERISTICS UP TO 33 MHz

(0°C to 70°C;  $V_{CC}=4.0$  to 5.5V)

PARAMETER	SYMBOL	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNITS	STRETCH
$\overline{RD}$ Pulse Width	$t_{RLRH}$	$2t_{CLCL}-11$ $t_{MCS}-11$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{WR}$ Pulse Width	$t_{WLWH}$	$2t_{CLCL}-11$ $t_{MCS}-11$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Valid Data In	$t_{RLDV}$		$2t_{CLCL}-25$ $t_{MCS}-25$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold After Read	$t_{RHDX}$	0		ns	
Data Float After Read	$t_{RHDZ}$		$t_{CLCL}-5$ $2t_{CLCL}-5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to Valid Data In	$t_{LLDV}$		$2.5t_{CLCL}-26$ $1.5t_{CLCL}-28+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to Valid Data In	$t_{AVDV1}$		$3t_{CLCL}-24$ $2t_{CLCL}-31+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to Valid Data In	$t_{AVDV2}$		$3.5t_{CLCL}-32$ $2.5t_{CLCL}-34+t_{MCS}$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$t_{LLWL}$	$0.5t_{CLCL}-5$ $1.5t_{CLCL}-5$	$0.5t_{CLCL}+6$ $1.5t_{CLCL}+8$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL1}$	$t_{CLCL}-9$ $2t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL2}$	$1.5t_{CLCL}-9$ $2.5t_{CLCL}-13$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Valid to $\overline{WR}$ Transition	$t_{QVWX}$	-9 $t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold After Write	$t_{WHQX}$	$t_{CLCL}-7$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Address Float	$t_{RLAZ}$		note 5	ns	
$\overline{RD}$ or $\overline{WR}$ High to ALE High	$t_{WHLH}$	0 $t_{CLCL}-5$	10 $t_{CLCL}+11$	ns	$t_{MCS}=0$ $t_{MCS}>0$

NOTE:  $t_{MCS}$  is a time period related to the Stretch memory cycle selection. The following table shows the value of  $t_{MCS}$  for each Stretch selection.

M2	M1	M0	MOVX CYCLES	$t_{MCS}$
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	$4 t_{CLCL}$
0	1	0	4 machine cycles	$8 t_{CLCL}$
0	1	1	5 machine cycles	$12 t_{CLCL}$
1	0	0	6 machine cycles	$16 t_{CLCL}$
1	0	1	7 machine cycles	$20 t_{CLCL}$
1	1	0	8 machine cycles	$24 t_{CLCL}$
1	1	1	9 machine cycles	$28 t_{CLCL}$

**EXTERNAL CLOCK CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Clock High Time	$t_{CHCX}$	10			ns	
Clock Low Time	$t_{CLCX}$	10			ns	
Clock Rise Time	$t_{CLCH}$			5	ns	
Clock Fall Time	$t_{CHCL}$			5	ns	

**SERIAL PORT MODE 0 TIMING CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Serial Port Clock Cycle Time SM2=0 12 clocks per cycle SM2=1 4 clocks per cycle	$t_{XLXL}$		$12t_{CLCL}$ $4t_{CLCL}$		ns	
Output Data Setup to Clock Rising Edge SM2=0 12 clocks per cycle SM2=1 4 clocks per cycle	$t_{QVXH}$		$10t_{CLCL}$ $3t_{CLCL}$		ns	
Output Data Hold from Clock Rising SM2=0 12 clocks per cycle SM2=1 4 clocks per cycle	$t_{XHGX}$		$2t_{CLCL}$ $t_{CLCL}$		ns	
Input Data Hold after Clock Rising SM2=0 12 clocks per cycle SM2=1 4 clocks per cycle	$t_{XHDX}$		$t_{CLCL}$ $t_{CLCL}$		ns	
Clock Rising Edge to Input Data Valid SM2=0 12 clocks per cycle SM2=1 4 clocks per cycle	$t_{XHDV}$		$11t_{CLCL}$ $3t_{CLCL}$		ns	

**EXPLANATION OF AC SYMBOLS**

In an effort to remain compatible with the original 8051 family, this device specifies the same parameter as such devices, using the same symbols. For completeness, the following is an explanation of the symbols.

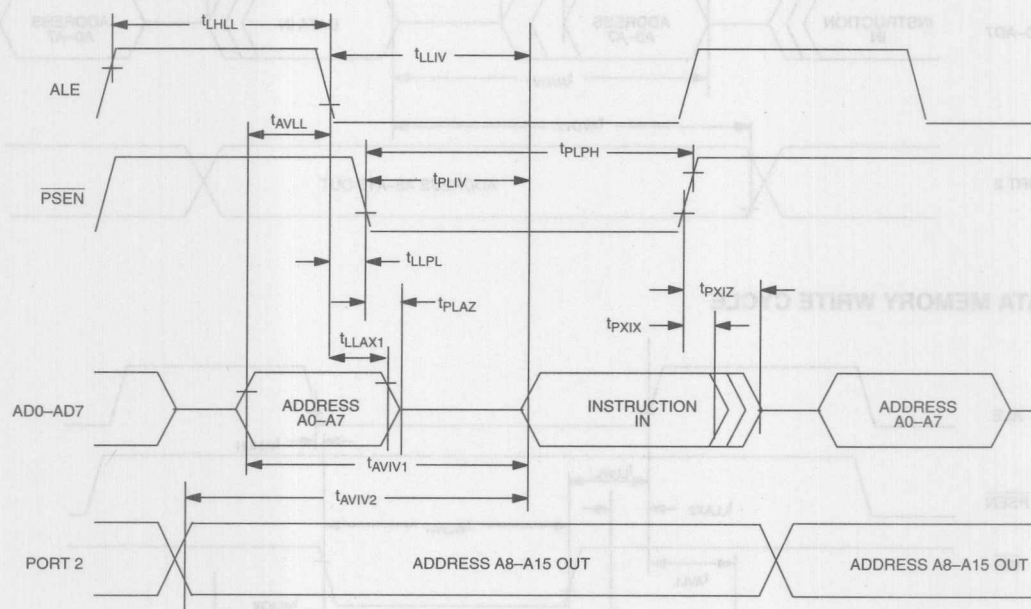
t	Time	L	Logic level low
A	Address	I	Instruction
C	Clock	P	PSEN
D	Input data	Q	Output data
H	Logic level high	R	RD signal
		V	Valid
		W	WR signal
		X	No longer a valid logic level
		Z	Tristate

**POWER CYCLE TIMING CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Crystal Start-up Time	$t_{CSU}$		1.8		ms	1
Power-on Reset Delay	$t_{POR}$			65536	$t_{CLCL}$	2

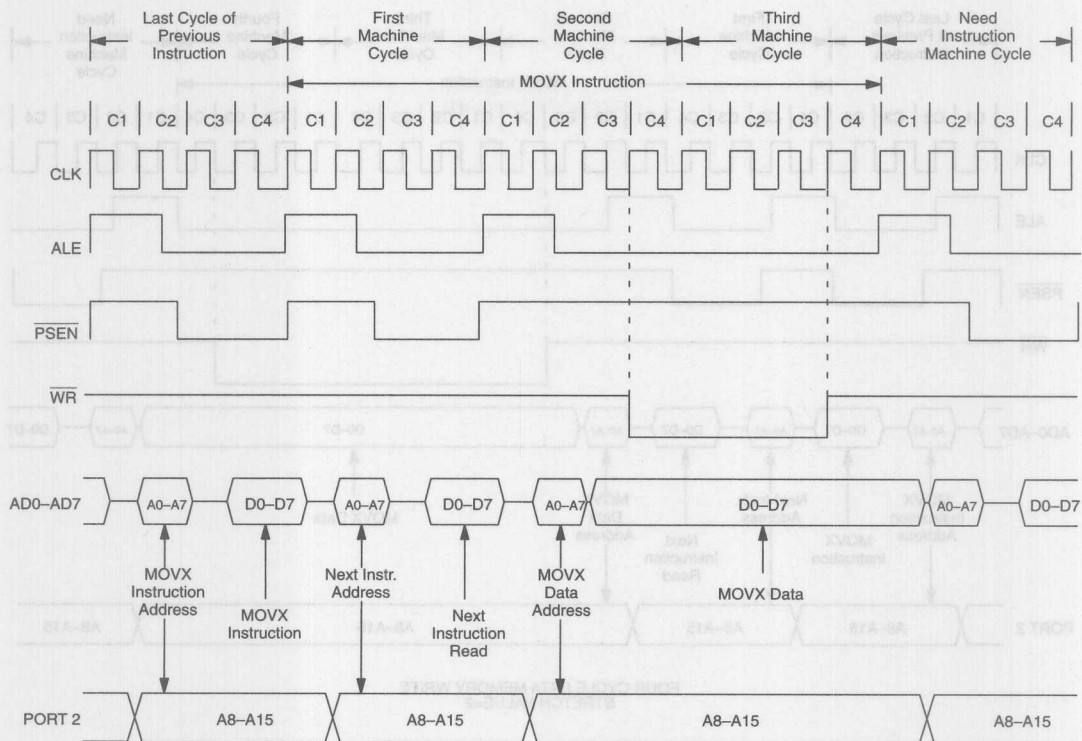
**NOTES FOR POWER CYCLE TIMING CHARACTERISTICS:**

1. Start-up time for crystals varies with load capacitance and manufacturer. Time shown is for an 11.0592 MHz crystal manufactured by Fox crystal.
2. Reset delay is a synchronous counter of crystal oscillations after crystal start-up. At 25 MHz, this time is 2.62 ms.

**PROGRAM MEMORY READ CYCLE**

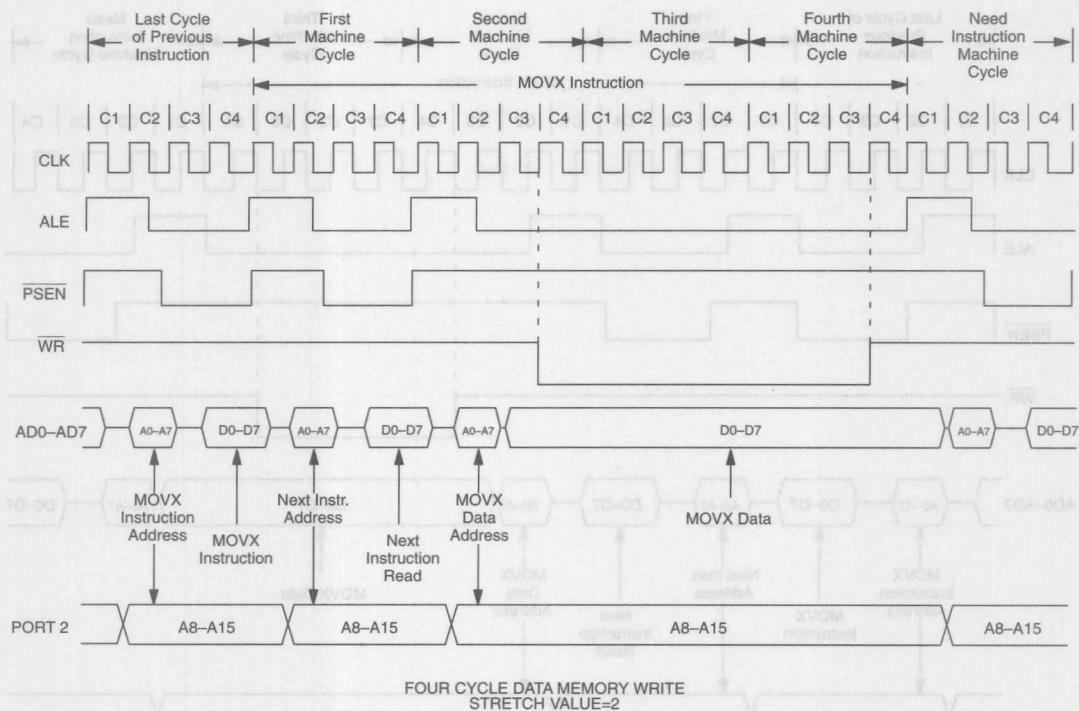


## DATA MEMORY WRITE WITH STRETCH=1

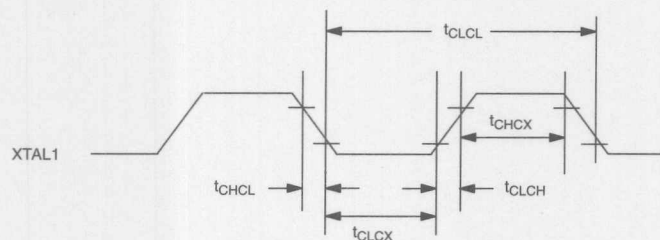




## DATA MEMORY WRITE WITH STRETCH=2

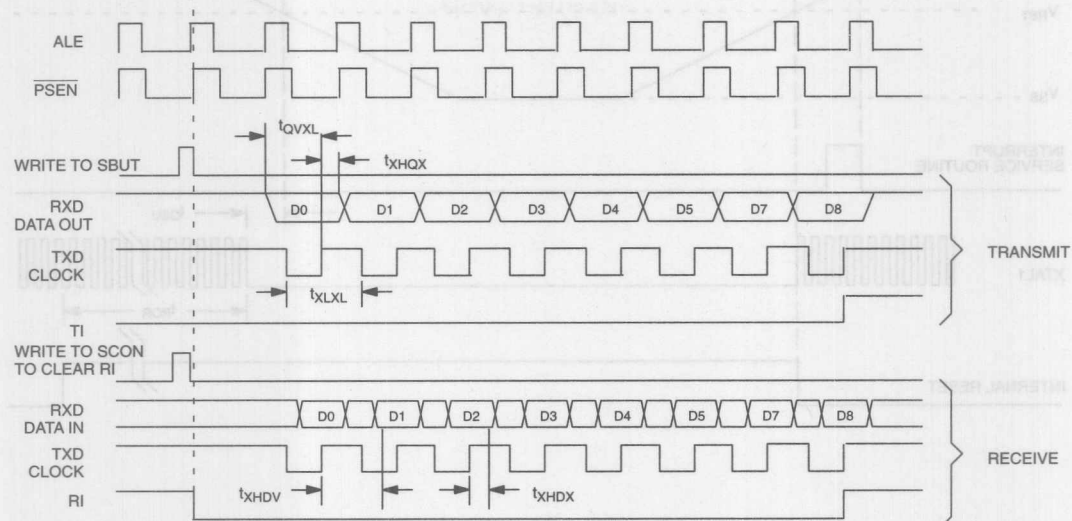


## EXTERNAL CLOCK DRIVE

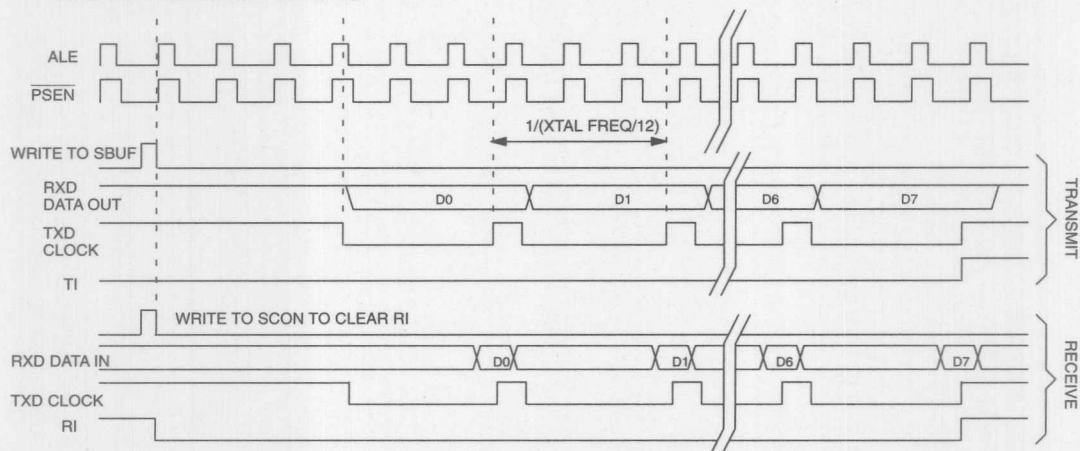


## SERIAL PORT MODE 0 TIMING

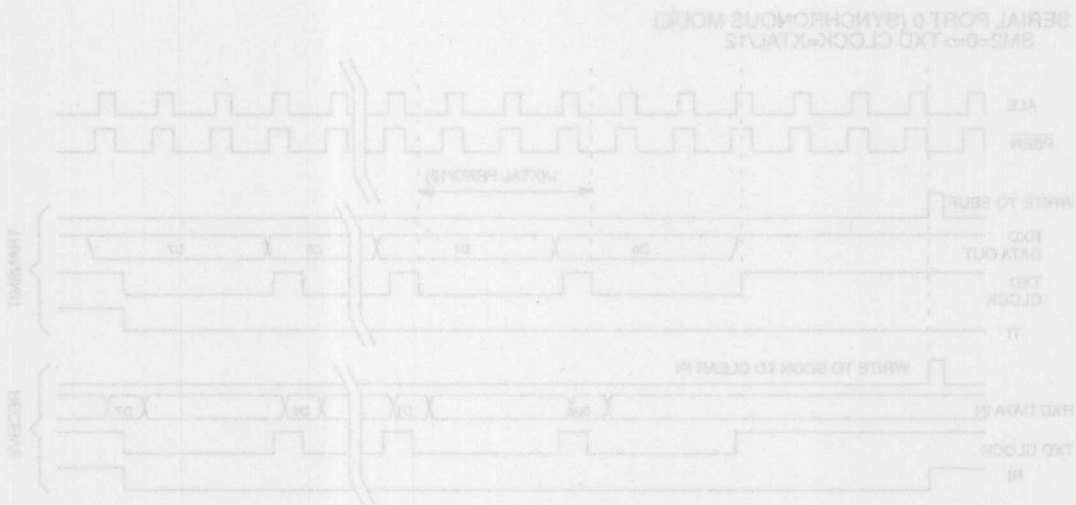
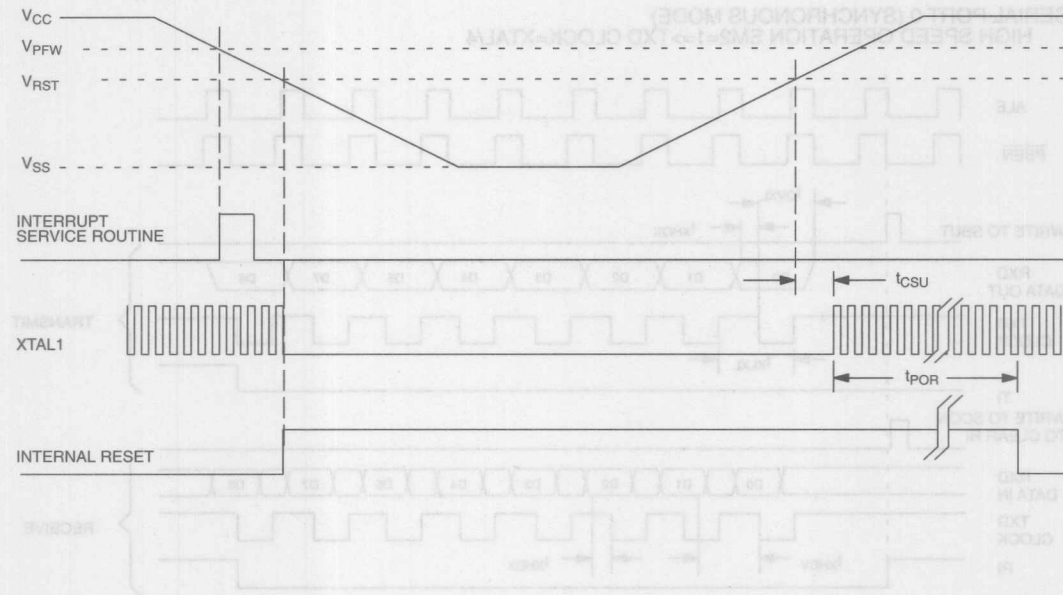
SERIAL PORT 0 (SYNCHRONOUS MODE)  
HIGH SPEED OPERATION  $SM2=1 \Rightarrow TXD\ CLOCK=XTAL/4$



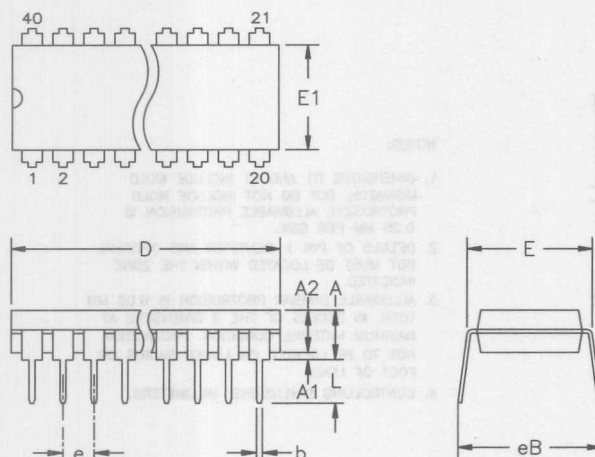
SERIAL PORT 0 (SYNCHRONOUS MODE)  
 $SM2=0 \Rightarrow TXD\ CLOCK=XTAL/12$



## POWER CYCLE TIMING



## 40-PIN PDIP (600 MIL)

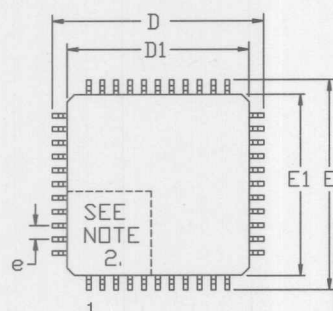


ALL DIMENSIONS ARE IN INCHES.

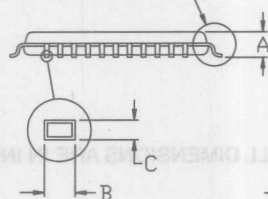
PKG	40-PIN	
DIM	MIN	MAX
A	—	0.200
A1	0.015	—
A2	0.140	0.160
b	0.014	0.022
c	0.008	0.012
D	1.980	2.085
E	0.600	0.625
E1	0.530	0.555
e	0.090	0.110
L	0.115	0.145
eB	0.600	0.700

56-G5000-000

## 40-PIN TQFP

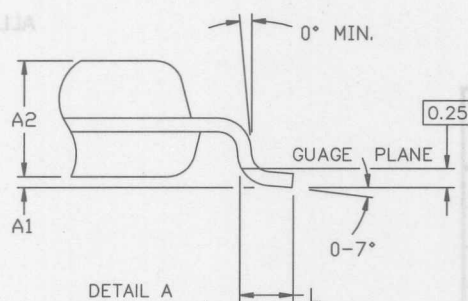


SEE DETAIL "A"



## NOTES:

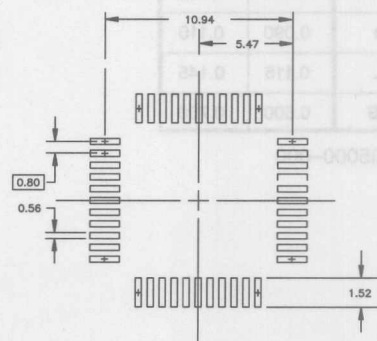
1. DIMENSIONS D1 AND E1 INCLUDE MOLD MISMATCH, BUT DO NOT INCLUDE MOLD PROTRUSION; ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE.
2. DETAILS OF PIN 1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATED.
3. ALLOWABLE DAMBAR PROTRUSION IS 0.08 MM TOTAL IN EXCESS OF THE B DIMENSION; AT MAXIMUM MATERIAL CONDITION. PROTRUSION NOT TO BE LOCATED ON LOWER RADIUS OR FOOT OF LEAD.
4. CONTROLLING DIMENSIONS: MILLIMETERS.



DETAIL A

PKG	44-PIN	
DIM	MIN	MAX
A	—	1.20
A1	0.05	0.15
A2	0.95	1.05
D	11.80	12.20
D1	10.00 BSC	
E	11.80	12.20
E1	10.00 BSC	
L	0.45	0.75
e	0.80 BSC	
B	0.30	0.45
C	0.09	0.20

56-G4012-001

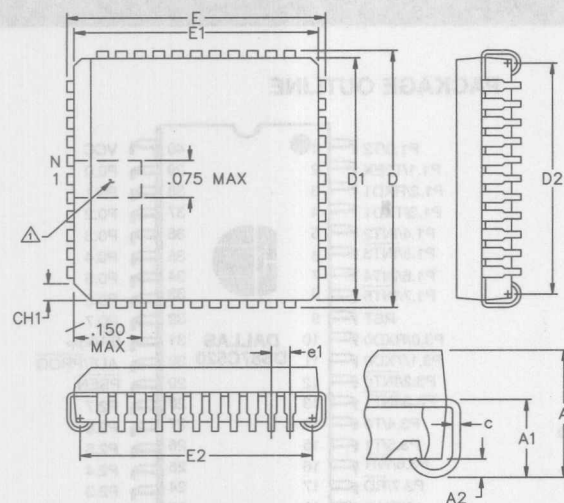
SUGGESTED PAD LAYOUT  
44 PIN TQFP, 10\*10\*1.0



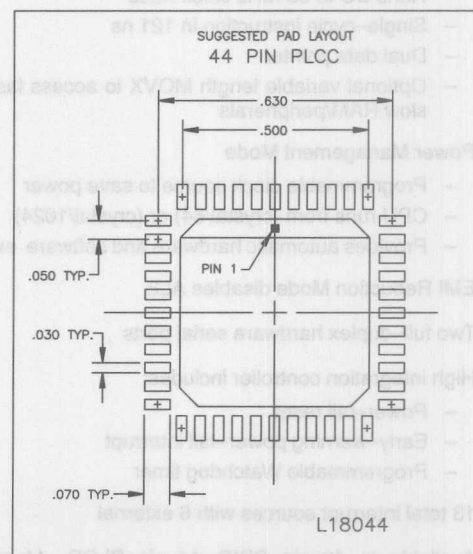
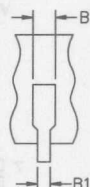
## 44-PIN PLCC

NOTE:

1. PIN-1 IDENTIFIER TO BE LOCATED IN ZONE INDICATED.
2. CONTROLLING DIMENSIONS ARE IN INCHES



PKG	44-PIN	
DIM	MIN	MAX
A	0.165	0.180
A1	0.090	0.120
A2	0.020	—
B	0.026	0.033
B1	0.013	0.021
c	0.009	0.012
CH1	0.042	0.048
D	0.685	0.695
D1	0.650	0.656
D2	0.590	0.630
E	0.685	0.695
E1	0.650	0.656
E2	0.590	0.630
e1	0.050 BSC	
N	44	—



56-G4003-001

# DALLAS

## SEMICONDUCTOR

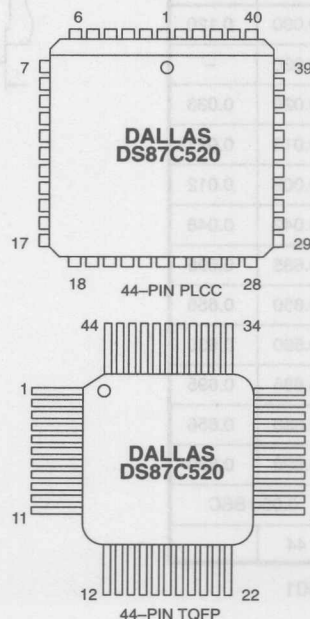
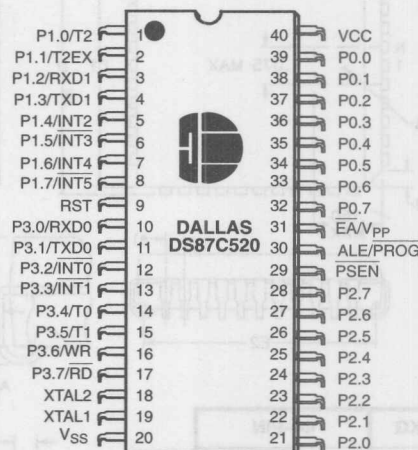
# DS87C520

## EPROM High-Speed Micro

### FEATURES

- 80C52 compatible
  - 8051 pin and instruction set compatible
  - Four 8-bit I/O ports
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
- Large On-chip Memory
  - 16KB EPROM (OTP)
  - 1KB extra on-chip SRAM for MOVX
- ROMSIZE™ Feature
  - Selects effective on-chip ROM size from 0 to 16KB
  - Allows access to entire external memory map
  - Dynamically adjustable by software
  - Useful as boot block for external FLASH
- High-Speed Architecture
  - 4 clocks/machine cycle (8051 = 12)
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM/peripherals
- Power Management Mode
  - Programmable clock source to save power
  - CPU runs from (crystal/64) or (crystal/1024)
  - Provides automatic hardware and software exit
- EMI Reduction Mode disables ALE
- Two full-duplex hardware serial ports
- High integration controller includes:
  - Power-fail reset
  - Early-warning power-fail interrupt
  - Programmable Watchdog timer
- 13 total interrupt sources with 6 external
- Available in 40-pin PDIP, 44-pin PLCC, 44-pin TQFP, and 40-pin windowed Cerdip

### PACKAGE OUTLINE



## DESCRIPTION

The DS87C520 is a fast 8051 compatible microcontroller. It features a redesigned processor core without wasted clock and memory cycles. As a result, it executes every 8051 instruction between 1.5 and 3 times faster than the original for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and the same crystal. The DS87C520 offers a maximum crystal speed of 33 MHz, resulting in apparent execution speeds of 82.5 MHz (approximately 2.5X).

The DS87C520 is pin compatible with all three packages of the standard 8051 and includes standard resources such as three timer/counters, serial port, and four 8-bit I/O ports. It features 16KB of EPROM with an extra 1KB of data RAM. Both OTP and windowed packages are available.

Besides greater speed, the micro includes a second full hardware serial port, seven additional interrupts, programmable watchdog timer, brown-out monitor, and

power-fail reset. The DS87C520 also provides dual data pointers (DPTRs) to speed block data memory moves. It also can adjust the speed of MOVX data memory access from two to nine machine cycles for flexibility in selecting external memory and peripherals.

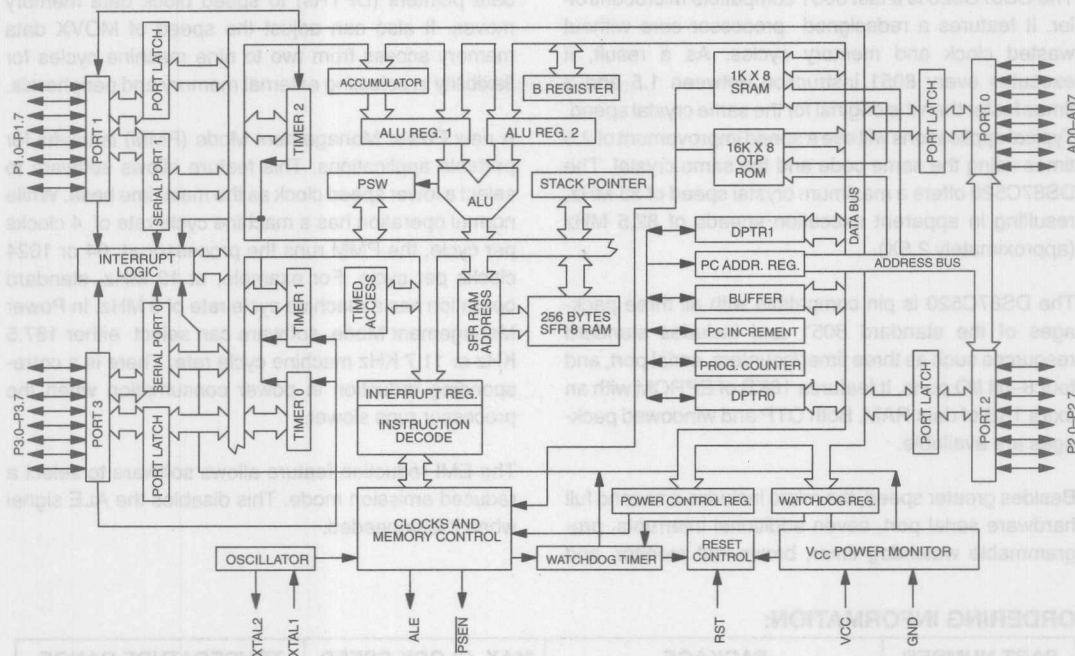
A new Power Management Mode (PMM) is useful for portable applications. This feature allows software to select a lower speed clock as the main time base. While normal operation has a machine cycle rate of 4 clocks per cycle, the PMM runs the processor at 64 or 1024 clocks per cycle. For example, at 12 MHz, standard operation has a machine cycle rate of 3 MHz. In Power Management Mode, software can select either 187.5 KHz or 11.7 KHz machine cycle rate. There is a corresponding reduction in power consumption when the processor runs slower.

The EMI reduction feature allows software to select a reduced emission mode. This disables the ALE signal when it is unneeded.

## ORDERING INFORMATION:

PART NUMBER	PACKAGE	MAX. CLOCK SPEED	TEMPERATURE RANGE
DS87C520-MCL	40-pin plastic DIP	33 MHz	0°C to 70°C
DS87C520-QCL	44-pin PLCC	33 MHz	0°C to 70°C
DS87C520-ECL	44-pin TQFP	33 MHz	0°C to 70°C
DS87C520-MNL	40-pin plastic DIP	33 MHz	-40°C to +85°C
DS87C520-QNL	44-pin PLCC	33 MHz	-40°C to +85°C
DS87C520-ENL	44-pin TQFP	33 MHz	-40°C to +85°C
DS87C520-WCL	40-pin windowed Cerdip	33 MHz	0°C to 70°C

DS87C520 BLOCK DIAGRAM Figure 1



PIN DESCRIPTION Table 1

DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION
40	44	38	V <sub>CC</sub>	V <sub>CC</sub> – +5V.
20	22, 23, 1	16, 17, 39	GND	GND – Digital circuit ground.
9	10	4	RST	<b>RST – Input.</b> The RST input pin contains a Schmitt voltage input to recognize external active high Reset inputs. The pin also employs an internal pull-down resistor to allow for a combination of wired OR external Reset sources. An RC is not required for power-up, as the DS87C520 provides this function internally.
18 19	20 21	14 15	XTAL2 XTAL1	<b>XTAL1, XTAL2 –</b> The crystal oscillator pins XTAL1 and XTAL2 provide support for parallel resonant, AT cut crystals. XTAL1 acts also as an input if there is an external clock source in place of a crystal. XTAL2 serves as the output of the crystal amplifier.
29	32	26	PSEN	<b>PSEN – Output.</b> The Program Store Enable output. This signal is commonly connected to optional external ROM memory as a chip enable. PSEN will provide an active low pulse and is driven high when external ROM is not being accessed.

DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION																		
30	33	27	ALE	<b>ALE – Output.</b> The Address Latch Enable output functions as a clock to latch the external address LSB from the multiplexed address/data bus on Port 0. This signal is commonly connected to the latch enable of an external 373 family transparent latch. ALE has a pulse width of 1.5 XTAL1 cycles and a period of four XTAL1 cycles. ALE is forced high when the DS87C520 is in a Reset condition. ALE can also be disabled using the EMI reduction mode.																		
39 38 37 36 35 34 33 32	43 42 41 40 39 38 37 36	37 36 35 34 33 32 31 30	P0.0 (AD0) P0.1 (AD1) P0.2 (AD2) P0.3 (AD3) P0.4 (AD4) P0.5 (AD5) P0.6 (AD6) P0.7 (AD7)	<b>Port 0 (AD0–7) – I/O.</b> Port 0 is an <u>open-drain</u> 8-bit bi-directional I/O port. As an alternate function Port 0 can function as the multiplexed address/data bus to access off-chip memory. During the time when ALE is high, the LSB of a memory address is presented. When ALE falls to a logic 0, the port transitions to a bi-directional data bus. This bus is used to read external ROM and read/write external RAM memory or peripherals. When used as a memory bus, the port provides active high drivers. The reset condition of Port 0 is tri-state. Pull-up resistors are required when using Port 0 as an I/O port.																		
1–8	2–9	40–44 1–3	P1.0–P1.7	<b>Port 1 – I/O.</b> Port 1 functions as both an 8-bit bi-directional I/O port and an alternate functional interface for Timer 2 I/O, new External Interrupts, and new Serial Port 1. The reset condition of Port 1 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C520 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes the output high (and input) state. The alternate modes of Port 1 are outlines as follows.  <table><tr><th>Port</th><th>Alternate Function</th></tr><tr><td>P1.0</td><td>T2 External I/O for Timer/Counter 2</td></tr><tr><td>P1.1</td><td>T2EX Timer/Counter 2 Capture/Reload Trigger</td></tr><tr><td>P1.2</td><td>RXD1 Serial Port 1 Input</td></tr><tr><td>P1.3</td><td>TXD1 Serial Port 1 Output</td></tr><tr><td>P1.4</td><td>INT2 External Interrupt 2 (Positive Edge Detect)</td></tr><tr><td>P1.5</td><td>INT3 External Interrupt 3 (Negative Edge Detect)</td></tr><tr><td>P1.6</td><td>INT4 External Interrupt 4 (Positive Edge Detect)</td></tr><tr><td>P1.7</td><td>INT5 External Interrupt 5 (Negative Edge Detect)</td></tr></table>	Port	Alternate Function	P1.0	T2 External I/O for Timer/Counter 2	P1.1	T2EX Timer/Counter 2 Capture/Reload Trigger	P1.2	RXD1 Serial Port 1 Input	P1.3	TXD1 Serial Port 1 Output	P1.4	INT2 External Interrupt 2 (Positive Edge Detect)	P1.5	INT3 External Interrupt 3 (Negative Edge Detect)	P1.6	INT4 External Interrupt 4 (Positive Edge Detect)	P1.7	INT5 External Interrupt 5 (Negative Edge Detect)
Port	Alternate Function																					
P1.0	T2 External I/O for Timer/Counter 2																					
P1.1	T2EX Timer/Counter 2 Capture/Reload Trigger																					
P1.2	RXD1 Serial Port 1 Input																					
P1.3	TXD1 Serial Port 1 Output																					
P1.4	INT2 External Interrupt 2 (Positive Edge Detect)																					
P1.5	INT3 External Interrupt 3 (Negative Edge Detect)																					
P1.6	INT4 External Interrupt 4 (Positive Edge Detect)																					
P1.7	INT5 External Interrupt 5 (Negative Edge Detect)																					
1 2 3 4 5 6 7 8	2 3 4 5 6 7 8 9	40 41 42 43 44 1 2 3																				



DIP	PLCC	TQFP	SIGNAL NAME	DESCRIPTION																		
21 22 23 24 25 26 27 28	24 25 26 27 28 29 30 31	18 19 20 21 22 23 24 25	P2.0 (A8) P2.1 (A9) P2.2 (A10) P2.3 (A11) P2.4 (A12) P2.5 (A13) P2.6 (A14) P2.7 (A15)	<b>Port 2 (A8–15) – I/O.</b> Port 2 is a bi-directional I/O port. The reset condition of Port 2 is logic high. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C520 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes both the output high and input state. As an alternate function Port 2 can function as MSB of the external address bus. This bus can be used to read external ROM and read/write external RAM memory or peripherals.																		
10–17	11, 13–19	5, 7–13	P3.0–P3.7	<b>Port 3 – I/O.</b> Port 3 functions as both an 8-bit bi-directional I/O port and an alternate functional interface for External Interrupts, Serial Port 0, Timer 0 and 1 Inputs, and RD and WR strobes. The reset condition of Port 3 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C520 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes both the output high and input state. The alternate modes of Port 3 are outlined below.  <table><tr><th>Port</th><th>Alternate Mode</th></tr><tr><td>P3.0</td><td>RXD0 Serial Port 0 Input</td></tr><tr><td>P3.1</td><td>TXD0 Serial Port 0 Output</td></tr><tr><td>P3.2</td><td>INT0 External Interrupt 0</td></tr><tr><td>P3.3</td><td>INT1 External Interrupt 1</td></tr><tr><td>P3.4</td><td>T0 Timer 0 External Input</td></tr><tr><td>P3.5</td><td>T1 Timer 1 External Input</td></tr><tr><td>P3.6</td><td>WR External Data Memory Write Strobe</td></tr><tr><td>P3.7</td><td>RD External Data Memory Read Strobe</td></tr></table>	Port	Alternate Mode	P3.0	RXD0 Serial Port 0 Input	P3.1	TXD0 Serial Port 0 Output	P3.2	INT0 External Interrupt 0	P3.3	INT1 External Interrupt 1	P3.4	T0 Timer 0 External Input	P3.5	T1 Timer 1 External Input	P3.6	WR External Data Memory Write Strobe	P3.7	RD External Data Memory Read Strobe
Port	Alternate Mode																					
P3.0	RXD0 Serial Port 0 Input																					
P3.1	TXD0 Serial Port 0 Output																					
P3.2	INT0 External Interrupt 0																					
P3.3	INT1 External Interrupt 1																					
P3.4	T0 Timer 0 External Input																					
P3.5	T1 Timer 1 External Input																					
P3.6	WR External Data Memory Write Strobe																					
P3.7	RD External Data Memory Read Strobe																					
31	35	29	EA	<b>EA – Input.</b> Connect to ground to force the DS87C520 to use an external ROM. The internal RAM is still accessible as determined by register settings. Connect EA to VCC to use internal ROM.																		
–	12 34	6 28	NC	<b>NC – Reserved.</b> These pins should not be connected. They are reserved for use with future devices in this family.																		

## COMPATIBILITY

The DS87C520 is a fully static CMOS 8051 compatible microcontroller designed for high performance. In most cases the DS87C520 can drop into an existing socket for the 80C51, 80C52, 87C51, or 87C52 to improve the operation significantly. While remaining familiar to 8051 family users, it has many new features. In general, software written for existing 8051 based systems works without modification on the DS87C520. The exception is critical timing since the High-Speed Micro performs its instructions much faster than the original for any given crystal selection. The DS87C520 runs the standard 8051 family instruction set and is pin compatible with DIP, PLCC or TQFP packages.

The DS87C520 provides three 16-bit timer/counters, full-duplex serial port (2), 256 bytes of direct RAM plus 1KB of extra MOVX RAM. I/O ports have the same operation as a standard 8051 product. Timers will default to a 12 clock per cycle operation to keep their timing compatible with original 8051 family systems. However, timers are individually programmable to run at the new 4 clocks per cycle if desired. The PCA is not supported.

The DS87C520 provides several new hardware features implemented by new Special Function Registers. A summary of these SFRs is provided below.

## PERFORMANCE OVERVIEW

The DS87C520 features a high speed 8051 compatible core. Higher speed comes not just from increasing the clock frequency, but from a newer, more efficient design.

This updated core does not have the dummy memory cycles that are present in a standard 8051. A conventional 8051 generates machine cycles using the clock frequency divided by 12. In the DS87C520, the same machine cycle takes four clocks. Thus the fastest instruction, 1 machine cycle, executes three times faster for the same crystal frequency. Note that these are identical instructions. The majority of instructions on the DS87C520 will see the full 3 to 1 speed improvement. Some instructions will get between 1.5 and 2.4 to 1 improvement. All instructions are faster than the original 8051.

The numerical average of all opcodes gives approximately a 2.5 to 1 speed improvement. Improvement of

individual programs will depend on the actual instructions used. Speed sensitive applications would make the most use of instructions that are three times faster. However, the sheer number of 3 to 1 improved opcodes makes dramatic speed improvements likely for any code. These architecture improvements and 0.8  $\mu$ m CMOS produce a peak instruction cycle in 121 ns (8.25 MIPs). The Dual Data Pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

## INSTRUCTION SET SUMMARY

All instructions in the DS87C520 perform the same functions as their 8051 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute and relative number of clocks.

For absolute timing of real-time events, the timing of software loops can be calculated using a table in the High-Speed Microcontroller User's Guide. However, counter/timers default to run at the older 12 clocks per increment. In this way, timer-based events occur at the standard intervals with software executing at higher speed. Timers optionally can run at 4 clocks per increment to take advantage of faster processor operation.

The relative time of two instructions might be different in the new architecture than it was previously. For example, in the original architecture, the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction used two machine cycles or 24 oscillator cycles. Therefore, they required the same amount of time. In the DS87C520, the MOVX instruction takes as little as two machine cycles or eight oscillator cycles but the "MOV direct, direct" uses three machine cycles or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times. This is because the DS87C520 usually uses one instruction cycle for each instruction byte. The user concerned with precise program timing should examine the timing of each instruction for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV. Refer to the High-Speed Microcontroller User's Guide for details and individual instruction timing.

## SPECIAL FUNCTION REGISTERS

Special Function Registers (SFRs) control most special features of the DS87C520. This allows the DS87C520 to have many new features but use the same instruction set as the 8051. When writing software to use a new feature, an equate statement defines the SFR to an assem-

bler or compiler. This is the only change needed to access the new function. The DS87C520 duplicates the SFRs contained in the standard 80C52. Table 2 shows the register addresses and bit locations. Many are standard 80C52 registers. The High-Speed Microcontroller User's Guide describes all SFRs.

**SPECIAL FUNCTION REGISTER LOCATIONS** Table 2

\* New functions are in bold

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
<b>DPL1</b>									84h
<b>DPH1</b>									85h
<b>DPS</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>SEL</b>	86h
PCON	SMOD_0	<b>SMOD0</b>	—	—	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/ $\bar{T}$	M1	M0	GATE	C/ $\bar{T}$	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
<b>CKCON</b>	<b>WD1</b>	<b>WD0</b>	<b>T2M</b>	<b>T1M</b>	<b>T0M</b>	<b>MD2</b>	<b>MD1</b>	<b>MD0</b>	8Eh
PORT1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
<b>EXIF</b>	<b>IE5</b>	<b>IE4</b>	<b>IE3</b>	<b>IE2</b>	<b>XT/RG</b>	<b>RGMD</b>	<b>RGSL</b>	<b>BGS</b>	91h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TL_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	—	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1	C0h
SBUF1	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0	C1h
ROMSIZE	—	—	—	—	—	RMS2	RMS1	RMS0	C2h
PMR	CD1	CD0	SWB	—	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C5h

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	—	—	—	—	—	—	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	D8h
ACC									E0h
EIE	—	—	—	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
EIP	—	—	—	PWDI	PX5	PX4	PX3	PX2	F8h

### MEMORY RESOURCES

Like the 8051, the DS87C520 uses three memory areas. These are program (ROM), data (RAM), and scratchpad RAM (registers). The DS87C520 contains on-chip quantities of all three areas.

The total memory configuration of the DS87C520 is 16KB of ROM, 1KB of data SRAM and 256 bytes of scratchpad or direct RAM. The 1KB of data space SRAM is read/write accessible and is memory mapped. This on-chip SRAM is reached by the MOVX instruction. It is not used for executable memory. The scratchpad area is 256 bytes of register mapped RAM and is identical to the RAM found on the 80C52. There is no conflict or overlap among the 256 bytes and the 1KB as they use different addressing modes and separate instructions.

### PROGRAM MEMORY ACCESS

On-chip ROM begins at address 0000h and is contiguous through 3FFFh (16KB). Exceeding the maximum address of on-chip ROM will cause the DS87C520 to access off-chip memory. However, the maximum on-chip decoded address is selectable by software using the ROMSIZE™ feature. Software can cause the DS87C520 to behave like a device with less on-chip memory. This is beneficial when overlapping external memory, such as Flash, is used.

The maximum memory size is dynamically variable. Thus a portion of memory can be removed from the memory map to access off-chip memory, then restored

to access on-chip memory. In fact, all of the on-chip memory can be removed from the memory map allowing the full 64KB memory space to be addressed from off-chip memory. ROM addresses that are larger than the selected maximum are automatically fetched from outside the part via Ports 0 and 2. A depiction of the ROM memory map is shown in Figure 2.

The ROMSIZE register is used to select the maximum on-chip decoded address for ROM. Bits RMS2, RMS1, RMS0 have the following affect.

RMS2	RMS1	RMS0	Maximum on-chip ROM Address
0	0	0	0KB
0	0	1	1KB
0	1	0	2KB
0	1	1	4KB
1	0	0	8KB
1	0	1	16KB (default)
1	1	0	Invalid – reserved
1	1	1	Invalid – reserved

The reset default condition is a maximum on-chip ROM address of 16KB. Thus no action is required if this feature is not used. When accessing external program memory, the first 16KB would be inaccessible. To select a smaller effective ROM size, software must alter bits RMS2–RMS0. Altering these bits requires a Timed Access procedure as explained later.

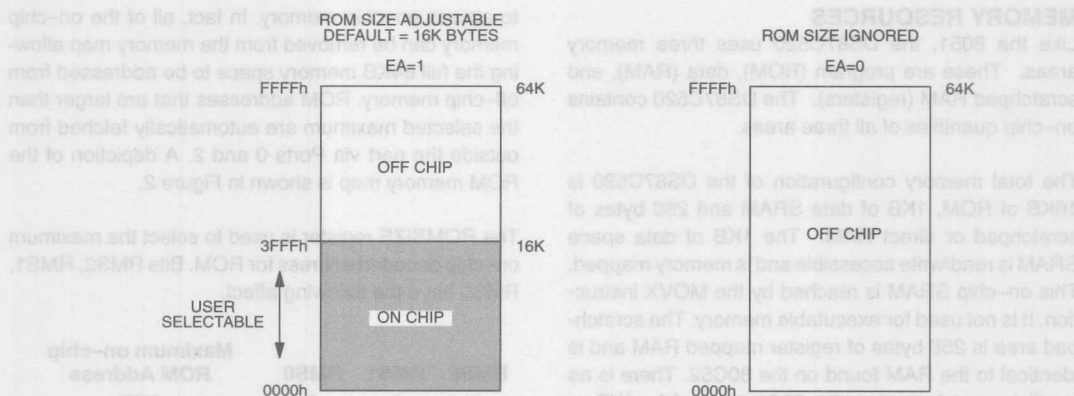


Care should be taken so that changing the ROMSIZE register does not corrupt program execution. For example, assume that a DS87C520 is executing instructions from internal program memory near the 12KB boundary (~3000h) and that the ROMSIZE register is currently configured for a 16KB internal program space. If software reconfigures the ROMSIZE register to 4KB (0000h–0FFFh) in the current state, the device will immediately jump to external program execution because program code from 4KB to 16KB (1000h–3FFFh) is no longer located on-chip. This could result in code misalignment and execution of an invalid instruction. The recommended method is to modify the ROMSIZE register from a location in memory that will be internal (or external) both before and after the operation. In the above example, the instruction which

modifies the ROMSIZE register should be located below the 4KB (1000h) boundary, so that it will be unaffected by the memory modification. The same precaution should be applied if the internal program memory size is modified while executing from external program memory.

Off-chip memory is accessed using the multiplexed address/data bus on P0 and the MSB address on P2. While serving as a memory bus, these pins are not I/O ports. This convention follows the standard 8051 method of expanding on-chip memory. Off-chip ROM access also occurs if the EA pin is a logic 0. EA overrides all bit settings. The PSEN signal will go active (low) to serve as a chip enable or output enable when Ports 0 and 2 fetch from external ROM.

**ROM MEMORY MAP** Figure 2



### DATA MEMORY ACCESS

Unlike many 8051 derivatives, the DS87C520 contains on-chip data memory. It also contains the standard 256 bytes of RAM accessed by direct instructions. These areas are separate. The MOVX instruction accesses the on-chip data memory. Although physically on-chip, software treats this area as though it was located off-chip. The 1KB of SRAM is between address 0000h and 03FFh.

Access to the on-chip data RAM is optional under software control. When enabled by software, the data SRAM is between 0000h and 03FFh. Any MOVX instruction that uses this area will go to the on-chip RAM while enabled. MOVX addresses greater than 03FFh automatically go to external memory through Ports 0 and 2.

When disabled, the 1KB memory area is transparent to the system memory map. Any MOVX directed to the space between 0000h and FFFFh goes to the expanded bus on Ports 0 and 2. This also is the default condition. This default allows the DS87C520 to drop into an existing system that uses these addresses for other hardware and still have full compatibility.

The on-chip data area is software selectable using two bits in the Power Management Register at location C4h. This selection is dynamically programmable. Thus access to the on-chip area becomes transparent to reach off-chip devices at the same addresses. The control bits are DME1 (PMR.1) and DME0 (PMR.0). They have the following operation:



DATA MEMORY ACCESS CONTROL Table 3

DME1	DME0	DATA MEMORY ADDRESS	MEMORY FUNCTION
0	0	0000h – FFFFh	External Data Memory *Default condition
0	1	0000h – 03FFh 0400h – FFFFh	Internal SRAM Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h – 03FFh 0400h – FFFBh FFFC h FFFDh – FFFFh	Internal SRAM Data Memory Reserved – no external access Read access to the status of lock bits Reserved – no external access

Notes on the status byte read at FFFCh with DME1, 0 = 1, 1: Bits 2–0 reflect the programmed status of the security lock bits LB2–LB0. They are individually set to a logic 1 to correspond to a security lock bit that has been programmed. These status bits allow software to verify that the part has been locked before running if desired. The bits are read only.

### STRETCH MEMORY CYCLE

The DS87C520 allows software to adjust the speed of off-chip data memory access. The micro is capable of performing the MOVX in as few as two instruction cycles. The on-chip SRAM uses this speed and any MOVX instruction directed internally uses two cycles. However, the time can be stretched for interface to external devices. This allows access to both fast memory and slow memory or peripherals with no glue logic. Even in high-speed systems, it may not be necessary or desirable to perform off-chip data memory access at full speed. In addition, there are a variety of memory mapped peripherals such as LCDs or UARTs that are slow.

The Stretch MOVX is controlled by the Clock Control Register at SFR location 8Eh as described below. It allows the user to select a Stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. A Stretch of seven will result in a MOVX of nine machine cycles. Software can dynamically change this value depending on the particular memory or peripheral.

On reset, the Stretch value will default to a one resulting in a three cycle MOVX for any external access. There-

fore, off-chip RAM access is not at full speed. This is a convenience to existing designs that may not have fast RAM in place. Internal SRAM access is always at full speed regardless of the Stretch setting. When desiring maximum speed, software should select a Stretch value of zero. When using very slow RAM or peripherals, select a larger Stretch value. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven causes the microcontroller to stretch the read/write strobe and all related timing. Also, setup and hold times are increased by one clock when using any Stretch greater than 0. This results in a wider read/write strobe and relaxed interface timing, allowing more time for memory/peripherals to respond. The timing of the variable speed MOVX is in the Electrical Specifications. Table 4 shows the resulting strobe widths for each Stretch value. The memory Stretch uses the Clock Control Special Function Register at SFR location 8Eh. The Stretch value is selected using bits CKCON.2–0. In the table, these bits are referred to as M2 through M0. The first Stretch (default) allows the use of common 120 ns RAMs without dramatically lengthening the memory access.

DATA MEMORY CYCLE STRETCH VALUES Table 4

CKCON.2-0			MEMORY CYCLES	RD OR WR STROBE WIDTH IN CLOCKS	STROBE WIDTH TIME @ 33 MHz
M2	M1	M0			
0	0	0	2 (forced internal)	2	60 ns
0	0	1	3 (default external)	4	121 ns
0	1	0	4	8	242 ns
0	1	1	5	12	364 ns
1	0	0	6	16	485 ns
1	0	1	7	20	606 ns
1	1	0	8	24	727 ns
1	1	1	9	28	848 ns

### DUAL DATA POINTER

The timing of block moves of data memory is faster using the DS87C520 Dual Data Pointer (DPTR). The standard 8051 DPTR is a 16-bit value that is used to address off-chip data RAM or peripherals. In the DS87C520, this data pointer is called DPTR0, located at SFR addresses 82h and 83h. These are the original locations. Using DPTR requires no modification of standard code. The new DPTR at SFR 84h and 85h is called DPTR1. The DPTR Select bit (DPS) chooses the active pointer. Its location is the lsb of the SFR location 86h. No other bits in register 86h have any effect and are 0. The user switches between data pointers by toggling the lsb of register 86h. The increment (INC) instruction is the fastest way to accomplish this. All DPTR-related instructions use the currently selected DPTR for any activity. Therefore it takes only one instruction to switch from a source to a destination address. Using the Dual Data Pointer saves code from needing to save source and destination addresses when doing a block move. The software simply switches between DPTR0 and 1 once software loads them. The relevant register locations are as follows:

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (lsb)

### POWER MANAGEMENT

Along with the standard Idle and power down (Stop) modes of the standard 80C52, the DS87C520 provides a new Power Management Mode. This mode allows the processor to continue functioning, yet to save power

compared with full operation. The DS87C520 also features several enhancements to Stop mode that make it more useful.

### POWER MANAGEMENT MODE (PMM)

Power Management Mode offers a complete scheme of reduced internal clock speeds that allow the CPU to run software but to use substantially less power. During default operation, the DS87C520 uses four clocks per machine cycle. Thus the instruction cycle rate is Clock/4. At 33 MHz crystal speed, the instruction cycle speed is 8.25 MHz (33/4). In PMM, the microcontroller continues to operate but uses an internally divided version of the clock source. This creates a lower power state without external components. It offers a choice of two reduced instruction cycle speeds (and two clock sources—discussed below). The speeds are (Clock/64) and (Clock/1024).

Software is the only mechanism to invoke the PMM. Table 5 illustrates the instruction cycle rate in PMM for several common crystal frequencies. Since power consumption is a direct function of operating speed, PMM 1 eliminates most of the power consumption while still allowing a reasonable speed of processing. PMM 2 runs very slow and provides the lowest power consumption without stopping the CPU. This is illustrated in Table 6.

Note that PMM provides a lower power condition than Idle mode. This is because in Idle mode, all clocked functions such as timers run at a rate of crystal divided by 4. Since wake-up from PMM is as fast as or faster than from Idle and PMM allows the CPU to operate (even if doing NOPs), there is little reason to use Idle mode in new designs.

**INSTRUCTION CYCLE RATE** Table 5

CRYSTAL SPEED	FULL OPERATION (4 CLOCKS)	PMM 1 (64 CLOCKS)	PMM 2 (1024 CLOCKS)
1.8432 MHz	460.8 KHz	28.8 KHz	1.8 KHz
11.0592 MHz	2.765 MHz	172.8 KHz	10.8 KHz
22 MHz	5.53 MHz	345.6 KHz	21.6 KHz
25 MHz	6.25 MHz	390.6 KHz	24.4 KHz
33 MHz	8.25 MHz	515.6 KHz	32.2 KHz

**OPERATING CURRENT ESTIMATES IN PMM** Table 6

CRYSTAL SPEED	FULL OPERATION (4 CLOCKS)	PMM 1 (64 CLOCKS)	PMM 2 (1024 CLOCKS)
1.8432 MHz	3.1 mA	1.2 mA	1.0 mA
3.57 MHz	5.3 mA	1.6 mA	1.1 mA
11.0592 MHz	15.5 mA	4.8 mA	4.0 mA
16 MHz	21 mA	7.1 mA	6.0 mA
22 MHz	25.5 mA	8.3 mA	6.5 mA
25 MHz	31 mA	9.7 mA	8.0 mA
33 MHz	36 mA	12.0 mA	10.0 mA

**CRYSTALESS PMM**

A major component of power consumption in PMM is the crystal amplifier circuit. The DS87C520 allows the user to switch CPU operation to an internal ring oscillator and turn off the crystal amplifier. The CPU would then have a clock source of approximately 2–4 MHz, divided by either 4, 64, or 1024. The ring is not accurate, so software can not perform precision timing. However, this mode allows an additional saving of between 0.5 and 6.0 mA depending on the actual crystal frequency. While this saving is of little use when running at 4 clocks per instruction cycle, it makes a major contribution when running in PMM1 or PMM2.

**PMM OPERATION**

Software invokes the PMM by setting the appropriate bits in the SFR area. The basic choices are divider speed and clock source. There are three speeds (4, 64, and 1024) and two clock sources (crystal and ring). Both the decisions and the controls are separate. Software will typically select the clock speed first. Then, it will perform the switch to ring operation if desired. Lastly, software can disable the crystal amplifier if desired.

There are two ways of exiting PMM. Software can remove the condition by reversing the procedure that invoked PMM or hardware can (optionally) remove it. To resume operation at a divide by 4 rate under software control, simply select 4 clocks per cycle, then crystal based operation if relevant. When disabling the crystal as the time base in favor of the ring oscillator, there are timing restrictions associated with restarting the crystal operation. Details are described below.

There are three registers containing bits that are concerned with PMM functions. They are Power Management Register (PMR; C4h), Status (STATUS; C5h), and External Interrupt Flag (EXIF; 91h)

**Clock Divider**

Software can select the instruction cycle rate by selecting bits CD1 (PMR.7) and CD0 (PMR.6) as follows:

CD1	CD0	Cycle rate
0	0	Reserved
0	1	4 clocks (default)
1	0	64 clocks
1	1	1024 clocks

The selection of instruction cycle rate will take effect after a delay of one instruction cycle. Note that the clock divider choice applies to all functions including timers. Since baud rates are altered, it will be difficult to conduct serial communication while in PMM. There are minor restrictions on accessing the clock selection bits. The processor must be running in a 4 clock state to select either 64 (PMM1) or 1024 (PMM2) clocks. This means software cannot go directly from PMM1 to PMM2 or visa versa. It must return to a 4 clock rate first.

### Switchback

To return to a 4 clock rate from PMM, software can simply select the CD1 and CD0 clock control bits to the 4 clocks per cycle state. However, the DS87C520 provides several hardware alternatives for automatic Switchback. If Switchback is enabled, then the DS87C520 will automatically return to a 4 clock per cycle speed when an interrupt occurs from an enabled, valid external interrupt source. A Switchback will also occur when a UART detects the beginning of a serial start bit if the serial receiver is enabled (REN=1). Note the beginning of a start bit does not generate an interrupt; this occurs on reception of a complete serial word. The automatic Switchback on detection of a start bit allows hardware to correct baud rates in time for a proper serial reception. A switchback will also occur when a byte is written to SBUF0 or SBUF1 for transmission.

Switchback is enabled by setting the SWB bit (PMR.5) to a 1 in software. For an external interrupt, Switchback will occur only if the interrupt source could really generate the interrupt. For example, if INT0 is enabled but has a low priority setting, then Switchback will not occur on INT0 if the CPU is servicing a high priority interrupt.

### Status

Information in the Status register assists decisions about switching into PMM. This register contains information about the level of active interrupts and the activity on the serial ports.

The DS87C520 supports three levels of interrupt priority. These levels are Power-fail, High, and Low. Bits

STATUS.7–5 indicate the service status of each level. If PIP (Power-fail Interrupt Priority; STATUS.7) is a 1, then the processor is servicing this level. If either HIP (High Interrupt Priority; STATUS.6) or LIP (Low Interrupt Priority; STATUS.5) is high, then the corresponding level is in service.

Software should not rely on a lower priority level interrupt source to remove PMM (Switchback) when a higher level is in service. Check the current priority service level before entering PMM. If the current service level locks out a desired Switchback source, then it would be advisable to wait until this condition clears before entering PMM.

Alternately, software can prevent an undesired exit from PMM by entering a low priority interrupt service level before entering PMM. This will prevent other low priority interrupts from causing a Switchback.

Status also contains information about the state of the serial ports. Serial Port Zero Receive Activity (SPRA0; STATUS.0) indicates a serial word is being received on Serial Port 0 when this bit is set to a 1. Serial Port Zero Transmit Activity (SPTA0; STATUS.1) indicates that the serial port is still shifting out a serial transmission. STATUS.2 and STATUS.3 provide the same information for Serial Port 1, respectively. These bits should be interrogated before entering PMM1 or PMM2 to ensure that no serial port operations are in progress. Changing the clock divisor rate during a serial transmission or reception will corrupt the operation.

### Crystal/Ring Operation

The DS87C520 allows software to choose the clock source as an independent selection from the instruction cycle rate. The user can select crystal-based or ring oscillator-based operation under software control. Power-on reset default is the crystal (or external clock) source. The ring may save power depending on the actual crystal speed. To save still more power, software can then disable the crystal amplifier. This process requires two steps. Reversing the process also requires two steps.



The  $XT/\overline{RG}$  bit (EXIF.3) selects the crystal or ring as the clock source. Setting  $XT/\overline{RG}=1$  selects the crystal. Setting  $XT/\overline{RG}=0$  selects the ring. The RGMD (EXIF.2) bit serves as a status bit by indicating the active clock source. RGMD=0 indicates the CPU is running from the crystal. RGMD=1 indicates it is running from the ring. When operating from the ring, disable the crystal amplifier by setting the XTOFF bit (PMR.3) to a 1. This can only be done when  $XT/\overline{RG}=0$ .

When changing the clock source, the selection will take effect after a one instruction cycle delay. This applies to changes from crystal to ring and vice versa. However, this assumes that the crystal amplifier is running. In most cases, when the ring is active, software previously disabled the crystal to save power. If ring operation is being used and the system must switch to crystal operation, the crystal must first be enabled. Set the XTOFF bit to a 0. At this time, the crystal oscillation will begin. The DS87C520 then provides a warm-up delay to make certain that the frequency is stable. Hardware will set the XTUP bit (STATUS.4) to a 1 when the crystal is ready for

use. Then software should write  $XT/\overline{RG}$  to a 1 to begin operating from the crystal. Hardware prevents writing  $XT/\overline{RG}$  to a 1 before  $XTUP=1$ . The delay between  $XTOFF=0$  and  $XTUP=1$  will be 65,536 crystal clocks in addition to the crystal cycle startup time.

Switchback has no effect on the clock source. If software selects a reduced clock divider and enables the ring, a Switchback will only restore the divider speed. The ring will remain as the time base until altered by software. If there is serial activity, Switchback usually occurs with enough time to create proper baud rates. This is not true if the crystal is off and the CPU is running from the ring. If sending a serial character that wakes the system from crystaless PMM, then it should be a dummy character of no importance with a subsequent delay for crystal startup.

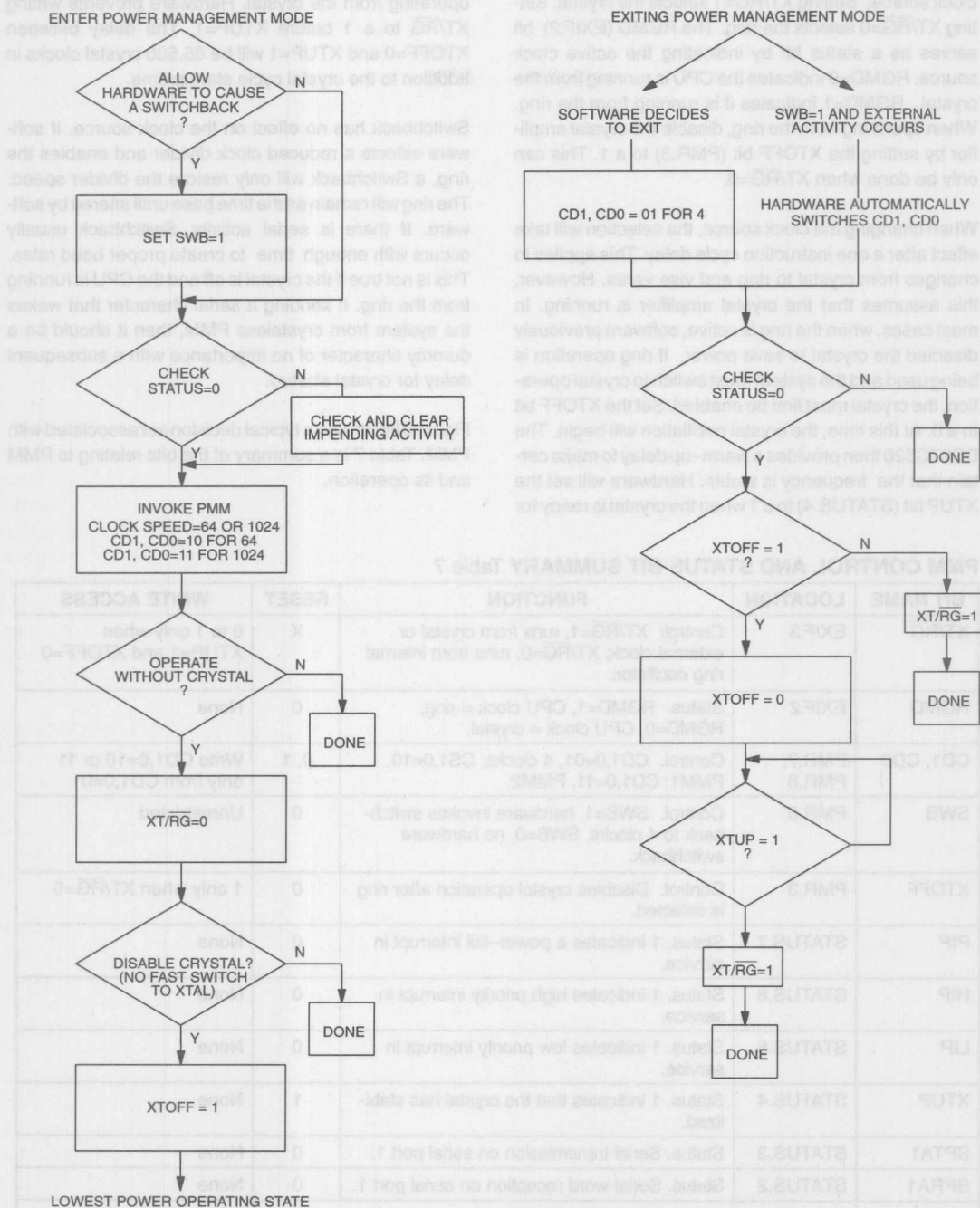
Figure 3 illustrates a typical decision set associated with PMM. Table 7 is a summary of the bits relating to PMM and its operation.

**PMM CONTROL AND STATUS BIT SUMMARY** Table 7

BIT NAME	LOCATION	FUNCTION	RESET	WRITE ACCESS
$XT/\overline{RG}$	EXIF.3	Control. $XT/\overline{RG}=1$ , runs from crystal or external clock; $XT/\overline{RG}=0$ , runs from internal ring oscillator.	X	0 to 1 only when $XTUP=1$ and $XTOFF=0$
RGMD	EXIF.2	Status. RGMD=1, CPU clock = ring; RGMD=0, CPU clock = crystal.	0	None
CD1, CD0	PMR.7, PMR.6	Control. CD1,0=01, 4 clocks; CS1,0=10, PMM1; CD1,0=11, PMM2.	0, 1	Write CD1,0=10 or 11 only from CD1,0=01
SWB	PMR.5	Control. SWB=1, hardware invokes switchback to 4 clocks, SWB=0, no hardware switchback.	0	Unrestricted
XTOFF	PMR.3	Control. Disables crystal operation after ring is selected.	0	1 only when $XT/\overline{RG}=0$
PIP	STATUS.7	Status. 1 indicates a power-fail interrupt in service.	0	None
HIP	STATUS.6	Status. 1 indicates high priority interrupt in service.	0	None
LIP	STATUS.5	Status. 1 indicates low priority interrupt in service.	0	None
XTUP	STATUS.4	Status. 1 indicates that the crystal has stabilized.	1	None
SPTA1	STATUS.3	Status. Serial transmission on serial port 1.	0	None
SPRA1	STATUS.2	Status. Serial word reception on serial port 1.	0	None
SPTA0	STATUS.1	Status. Serial transmission on serial port 0.	0	None
SPRA0	STATUS.0	Status. Serial word reception on serial port 0.	0	None



# INVOKING AND CLEARING PMM Figure 3



## IDLE MODE

Setting the lsb of the Power Control register (PCON; 87h) invokes the Idle mode. Idle will leave internal clocks, serial ports and timers running. Power consumption drops because the CPU is not active. Since clocks are running, the Idle power consumption is a function of crystal frequency. It should be approximately 1/2 of the operational power at a given frequency. The CPU can exit the Idle state with any interrupt or a reset. Idle is available for backward software compatibility. The system can now reduce power consumption to below Idle levels by using PMM1 or PMM2 and running NOPs.

## STOP MODE ENHANCEMENTS

Setting bit 1 of the Power Control register (PCON; 87h) invokes the Stop mode. Stop mode is the lowest power state since it turns off all internal clocking. The  $I_{CC}$  of a standard Stop mode is approximately 1  $\mu$ A (but is specified in the Electrical Specifications). The CPU will exit Stop mode from an external interrupt or a reset condition. Internally generated interrupts (timer, serial port, watchdog) are not useful since they require clocking activity.

The DS87C520 provides two enhancements to the Stop mode. As documented below, the DS87C520 provides a band-gap reference to determine Power-fail Interrupt and Reset thresholds. The default state is that the band-gap reference is off while in Stop mode. This allows the extremely low power state mentioned above. A user can optionally choose to have the band-gap enabled during Stop mode. With the band-gap reference enabled, PFI and Power-fail Reset are functional and are a valid means for leaving Stop mode. This allows software to detect and compensate for a brown-out or power supply sag, even when in Stop mode. In Stop mode with the band-gap enabled,  $I_{CC}$  will be approximately 50  $\mu$ A compared with 1  $\mu$ A with the band-gap off. If a user does not require a Power-fail Reset or Interrupt while in Stop mode, the band-gap can remain disabled. Only the most power sensitive applications should turn off the band-gap, as this results in an uncontrolled power-down condition.

The control of the band-gap reference is located in the Extended Interrupt Flag register (EXIF; 91h). Setting BGS (EXIF.0) to a 1 will keep the band-gap reference enabled during Stop mode. The default or reset condi-

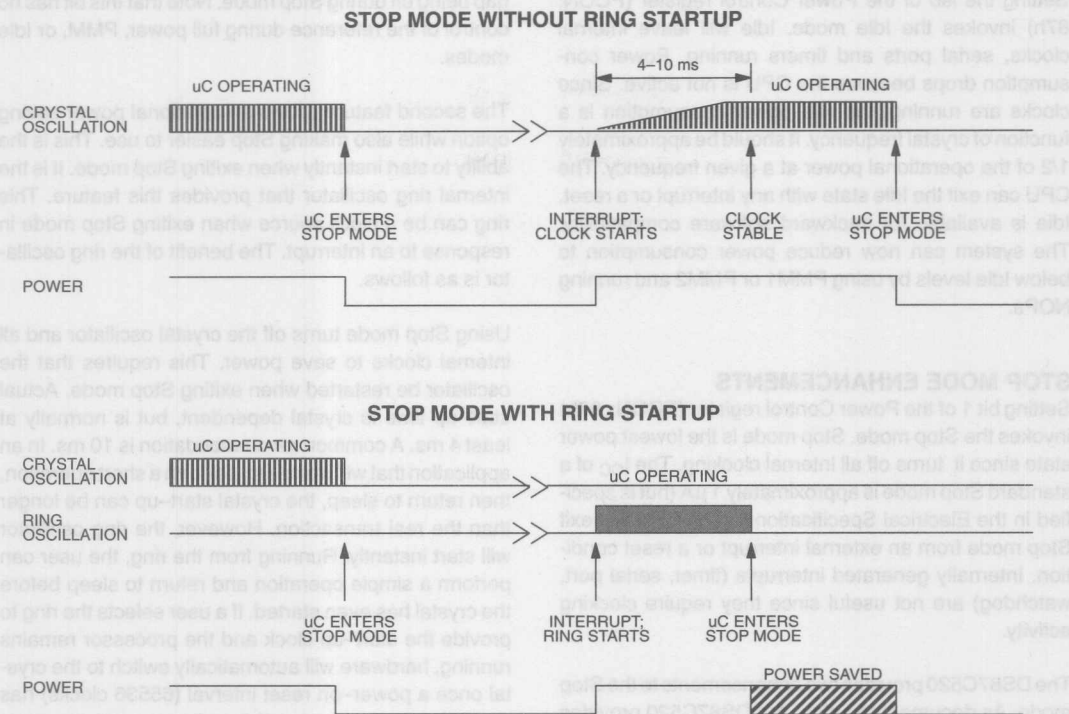
tion is with the bit at a logic 0. This results in the band-gap being off during Stop mode. Note that this bit has no control of the reference during full power, PMM, or Idle modes.

The second feature allows an additional power saving option while also making Stop easier to use. This is the ability to start instantly when exiting Stop mode. It is the internal ring oscillator that provides this feature. This ring can be a clock source when exiting Stop mode in response to an interrupt. The benefit of the ring oscillator is as follows.

Using Stop mode turns off the crystal oscillator and all internal clocks to save power. This requires that the oscillator be restarted when exiting Stop mode. Actual start-up time is crystal dependent, but is normally at least 4 ms. A common recommendation is 10 ms. In an application that will wake-up, perform a short operation, then return to sleep, the crystal start-up can be longer than the real transaction. However, the ring oscillator will start instantly. Running from the ring, the user can perform a simple operation and return to sleep before the crystal has even started. If a user selects the ring to provide the start-up clock and the processor remains running, hardware will automatically switch to the crystal once a power-on reset interval (65536 clocks) has expired. Hardware uses this value to assure proper crystal start even though power is not being cycled.

The ring oscillator runs at approximately 2–4 MHz but will not be a precise value. Do not conduct real-time precision operations (including serial communication) during this ring period. Figure 3 shows how the operation would compare when using the ring, and when starting up normally. The default state is to exit Stop mode without using the ring oscillator.

The RGSL – Ring Select bit at EXIF.1 (EXIF; 91h) controls this function. When RGSL=1, the CPU will use the ring oscillator to exit Stop mode quickly. As mentioned above, the processor will automatically switch from the ring to the crystal after a delay of 65,536 crystal clocks. For a 3.57 MHz crystal, this is approximately 18 ms. The processor sets a flag called RGMD–Ring Mode, located at EXIF.2, that tells software that the ring is being used. The bit will be a logic 1 when the ring is in use. Attempt no serial communication or precision timing while this bit is set, since the operating frequency is not precise.

**RING OSCILLATOR EXIT FROM STOP MODE** Figure 4

Note: Diagram assumes that the operation following Stop requires less than 18 ms to complete.

**EMI REDUCTION**

The DS87C520 allows software to reduce EMI. One of the major contributors to radiated noise in an 8051 based system is the toggling of ALE. The DS87C520 allows software to disable ALE when not used by setting the ALE OFF (PMR.2) bit to a 1. When ALEOFF=1, ALE will still toggle during an off-chip MOVX. However, ALE will remain in a static mode when performing on-chip memory access. The default state of ALEOFF=0 so ALE toggles at a frequency of XTAL/4.

**PERIPHERAL OVERVIEW**

The DS87C520 provides several of the most commonly needed peripheral functions in microcomputer-based systems. These new functions include a second serial port, Power-fail Reset, Power-fail Interrupt, and a programmable Watchdog Timer. These are described below, and more details are available in the High-Speed Microcontroller User's Guide.

**SERIAL PORTS**

The DS87C520 provides a serial port (UART) that is identical to the 80C52. In addition it includes a second hardware serial port that is a full duplicate of the standard one. This port optionally uses pins P1.2 (RXD1) and P1.3 (TXD1). It has duplicate control functions included in new SFR locations.

Both ports can operate simultaneously but can be at different baud rates or even in different modes. The second serial port has similar control registers (SCON1 at C0h, SBUF1 at C1h) to the original. The new serial port can only use Timer 1 for timer generated baud rates.

**TIMER RATE CONTROL**

There is one important difference between the DS87C520 and 8051 regarding timers. The original 8051 used 12 clocks per cycle for timers as well as for

machine cycles. The DS87C520 architecture normally uses four clocks per machine cycle. However, in the area of timers and serial ports, the DS87C520 will default to 12 clocks per cycle on reset. This allows existing code with real-time dependencies such as baud rates to operate properly.

If an application needs higher speed timers or serial baud rates, the user can select individual timers to run at the 4 clock rate. The Clock Control register (CKCON; 8Eh) determines these timer speeds. When the relevant CKCON bit is a logic 1, the DS87C520 uses four clocks per cycle to generate timer speeds. When the bit is a 0, the DS87C520 uses 12 clocks for timer speeds. The reset condition is a 0. CKCON.5 selects the speed of Timer 2. CKCON.4 selects Timer 1 and CKCON.3 selects Timer 0. Unless a user desires very fast timing, it is unnecessary to alter these bits. Note that the timer controls are independent.

### POWER FAIL RESET

The DS87C520 uses a precision band-gap voltage reference to decide if  $V_{CC}$  is out of tolerance. While powering up, the internal monitor circuit maintains a reset state until  $V_{CC}$  rises above the  $V_{RST}$  level. Once above this level, the monitor enables the crystal oscillator and counts 65536 clocks. It then exits the reset state. This power-on reset (POR) interval allows time for the oscillator to stabilize.

A system needs no external components to generate a power-related reset. Anytime  $V_{CC}$  drops below  $V_{RST}$ , as in power failure or a power drop, the monitor will generate and hold a reset. It occurs automatically, needing no action from the software. Refer to the Electrical Specifications for the exact value of  $V_{RST}$ .

### POWER FAIL INTERRUPT

The voltage reference that sets a precise reset threshold also generates an optional early warning Power-Fail Interrupt (PFI). When enabled by software, the processor will vector to program memory address 0033h if  $V_{CC}$  drops below  $V_{PFW}$ . PFI has the highest priority. The PFI enable is in the Watchdog Control SFR (WDCON – D8h). Setting WDCON.5 to a logic 1 will enable the PFI.

Application software can also read the PFI flag at WDCON.4. A PFI condition sets this bit to a 1. The flag is independent of the interrupt enable and software must manually clear it.

### WATCHDOG TIMER

To prevent software from losing control, the DS87C520 includes a programmable Watchdog Timer. The Watchdog is a free running timer that sets a flag if allowed to reach a preselected time-out. It can be (re)started by software.

A typical application is to select the flag as a reset source. When the Watchdog times out, it sets its flag which generates reset. Software must restart the timer before it reaches its time-out or the processor is reset.

Software can select one of four time-out values. Then, it restarts the timer and enables the reset function. After enabling the reset function, software must then restart the timer before its expiration or hardware will reset the CPU. Both the Watchdog Reset Enable and the Watchdog Restart control bits are protected by a "Timed Access" circuit. This prevents errant software from accidentally clearing the Watchdog. Time-out values are precise since they are a function of the crystal frequency as shown in Table 8. For reference, the time periods at 33 MHz also are shown.

The Watchdog also provides a useful option for systems that do not require a reset circuit. It will set an interrupt flag 512 clocks before setting the reset flag. Software can optionally enable this interrupt source. The interrupt is independent of the reset. A common use of the interrupt is during debug, to show developers where the Watchdog times out. This indicates where the Watchdog must be restarted by software. The interrupt also can serve as a convenient time-base generator or can wake-up the processor from power saving modes.

The Watchdog function is controlled by the Clock Control (CKCON – 8Eh), Watchdog Control (WDCON – D8h), and Extended Interrupt Enable (EIE – E8h) SFRs. CKCON.7 and CKCON.6 are WD1 and WD0 respectively and they select the Watchdog time-out period as shown in Table 8.



**WATCHDOG TIME-OUT VALUES** Table 8

WD1	WD0	INTERRUPT TIME-OUT	TIME (33 MHz)	RESET TIME-OUT	TIME (33 MHz)
0	0	$2^{17}$ clocks	3.9718 ms	$2^{17} + 512$ clocks	3.9874 ms
0	1	$2^{20}$ clocks	31.77 ms	$2^{20} + 512$ clocks	31.79 ms
1	0	$2^{23}$ clocks	254.20 ms	$2^{23} + 512$ clocks	254.21 ms
1	1	$2^{26}$ clocks	2033.60 ms	$2^{26} + 512$ clocks	2033.62 ms

As shown above, the Watchdog Timer uses the crystal frequency as a time base. A user selects one of four counter values to determine the time-out. These clock counter lengths are  $2^{17}=131,072$  clocks;  $2^{20}=1,048,576$ ;  $2^{23}=8,388,608$  clocks; and  $2^{26}=67,108,864$  clocks. The times shown in Table 8 above are with a 33 MHz crystal frequency. Once the counter chain has completed a full interrupt count, hardware will set an interrupt flag. Regardless of whether the user enables this interrupt, there are then 512 clocks left until the reset flag is set. Software can enable the interrupt and reset individually. Note that the Watchdog is a free running timer and does not require an enable. There are five control bits in special function registers that affect the Watchdog Timer and two status flags that report to the user. WDIF (WDCON.3) is the interrupt flag that is set at timer termination when there are 512 clocks remaining until the reset flag is set. WTRF (WDCON.2) is the flag that is set when the timer has completely timed out. This flag is normally associated with a CPU reset and allows software to determine the reset source.

EWT (WDCON.1) is the enable for the Watchdog timer reset function. RWT (WDCON.0) is the bit that software uses to restart the Watchdog Timer. Setting this bit restarts the timer for another full interval. Application software must set this bit before the time-out. Both of these bits are protected by Timed Access discussed below. As mentioned previously, WD1 and 0 (CKCON.7 and 6) select the time-out. Finally, the user can enable the Watchdog Interrupt using EWDI (EIE.4). The Special Function Register map is shown above.

## INTERRUPTS

The DS87C520 provides 13 interrupt sources with three priority levels. The Power-fail Interrupt (PFI) has the highest priority. Software can assign high or low priority to other sources. All interrupts that are new to the 8051 family, except for the PFI, have a lower natural priority than the originals.

**INTERRUPT SOURCES AND PRIORITIES** Table 9

NAME	DESCRIPTION	VECTOR	NATURAL PRIORITY	8051/DALLAS
PFI	Power Fail Interrupt	33h	1	DALLAS
INT0	External Interrupt 0	03h	2	8051
TF0	Timer 0	0Bh	3	8051
INT1	External Interrupt 1	13h	4	8051
TF1	Timer 1	1Bh	5	8051
SCON0	TI0 or RI0 from serial port 0	23h	6	8051
TF2	Timer 2	2Bh	7	8051
SCON1	TI1 or RI1 from serial port 1	3Bh	8	DALLAS
INT2	External Interrupt 2	43h	9	DALLAS
INT3	External Interrupt 3	4Bh	10	DALLAS
INT4	External Interrupt 4	53h	11	DALLAS
INT5	External Interrupt 5	5Bh	12	DALLAS
WDTI	Watchdog Time-Out Interrupt	63h	13	DALLAS



## TIMED ACCESS PROTECTION

It is useful to protect certain SFR bits from an accidental write operation. The Timed Access procedure stops an errant CPU from accidentally changing these bits. It requires that the following instructions precede a write of a protected bit.

```
MOV    0C7h, #0AAh
MOV    0C7h, #55h
```

Writing an AAh then a 55h to the Timed Access register (location C7h) opens a 3 cycle window for write access. The window allows software to modify a protected bit(s). If these instructions do not immediately precede the write operation, then the write will not take effect. The protected bits are:

EXIF.0	BGS	Band-gap Select
WDCON.6	POR	Power-on Reset flag
WDCON.1	EWT	Enable Watchdog Reset
WDCON.0	RWT	Restart Watchdog
WDCON.3	WDIF	Watchdog Interrupt Flag
ROMSIZE.2	RMS2	ROM size select 2
ROMSIZE.1	RMS1	ROM size select 1
ROMSIZE.0	RMS0	ROM size select 0

## EPROM PROGRAMMING

The DS87C520 follows standards for a 16KB EPROM version in the 8051 family. It is available in a UV erasable, ceramic windowed package and in plastic packages for one-time user-programmable versions. The part has unique signature information so programmers can support its specific EPROM options.

Most commercially available device programmers will directly support Dallas Semiconductor microcontrollers. If your programmer does not, please contact the manufacturer for updated software.

## PROGRAMMING PROCEDURE

The DS87C520 should run from a clock speed between 4 and 6 MHz when programmed. The programming fixture should apply address information for each byte to the address lines and the data value to the data lines. The control signals must be manipulated as shown in Table 10. The diagram in Figure 5 shows the expected electrical connection for programming. Note that the programmer must apply addresses in demultiplexed fashion to Ports 1 and 2 with data on Port 0. Waveforms and timing are provided in the Electrical Specifications.

Program the DS87C520 as follows:

1. Apply the address value,
2. Apply the data value,
3. Select the programming option from Table 10 using the control signals,
4. Increase the voltage on  $V_{PP}$  from 5V to 12.75V if writing to the EPROM,
5. Pulse the  $\overline{PROG}$  signal five times for EPROM array and 25 times for encryption table, lock bits, and other EPROM bits,
6. Repeat as many times as necessary.

**EPROM PROGRAMMING MODES** Table 10

MODE		RST	PSEN	ALE/PROG	EA/VPP	P2.6	P2.7	P3.3	P3.6	P3.7
Program Code Data		H	L	PL	12.75V	L	H	H	H	H
Verify Code Data		H	L	H	H	L	L	L	H	H
Program Encryption Array Address 0–3Fh		H	L	PL	12.75V	L	H	H	L	H
Program Lock Bits	LB1	H	L	PL	12.75V	H	H	H	H	H
	LB2	H	L	PL	12.75V	H	H	H	L	L
	LB3	H	L	PL	12.75V	H	L	H	H	L
Program Option Register Address FCh		H	L	PL	12.75V	L	H	H	L	L
Read Signature or Option Registers 30, 31, 60, FCh		H	L	H	H	L	L	L	L	L

\* PL indicates pulse to a logic low.

**EPROM LOCK BITS** Table 11

LEVEL	LOCK BITS			PROTECTION
	LB1	LB2	LB3	
1	U	U	U	No program lock. Encrypted verify if encryption table was programmed.
2	P	U	U	Prevent MOV <sub>C</sub> instructions in external memory from reading program bytes in internal memory. EA is sampled and latched on reset. Allow no further programming of EPROM.
3	P	P	U	Level 2 plus no verify operation. Also, prevent MOV <sub>X</sub> instructions in external memory from reading SRAM (MOV <sub>X</sub> ) in internal memory.
4	P	P	P	Level 3 plus no external execution.

### SECURITY OPTIONS

The DS87C520 employs a standard three-level lock that restricts viewing of the EPROM contents. A 64-byte Encryption Array allows the authorized user to verify memory by presenting the data in encrypted form.

#### Lock Bits

The security lock consists of three lock bits. These bits select a total of four levels of security. Higher levels provide increasing security but also limit application flexibility. Table 11 shows the security settings. Note that the programmer cannot directly read the state of the security lock. User software has access to this information as described in the Memory section.

#### Encryption Array

The Encryption Array allows an authorized user to verify EPROM without allowing the true memory to be dumped. During a verify, each byte is Exclusive NORed (XNOR) with a byte in the Encryption Array. This results in a true representation of the EPROM while the Encryption is unprogrammed (FFh). Once the Encryption Array is programmed in a non-FFh state, the verify value will be encrypted.

For encryption to be effective, the Encryption Array must be unknown to the party that is trying to verify memory. The entire EPROM also should be a non-FFh state or the Encryption Array can be discovered.

The Encryption Array is programmed as shown in Table 10. Note that the programmer cannot read the array. Also note that the verify operation always uses the

Encryption Array. The array has no impact while FFh. Simply programming the array to a non-FFh state will cause the encryption to function.

### OTHER EPROM OPTIONS

The DS87C520 has user selectable options that must be set before beginning software execution. These options use EPROM bits rather than SFRs.

Program the EPROM selectable options as shown in Table 10. The Option Register sets or reads these selections. The bits in the Option Control Register have the following function:

Bit 7–4 Reserved, program to a 1.

Bit 3 Watchdog POR default. Set=1; Watchdog reset function is disabled on power-up. Set=0; Watchdog reset function is enabled automatically.

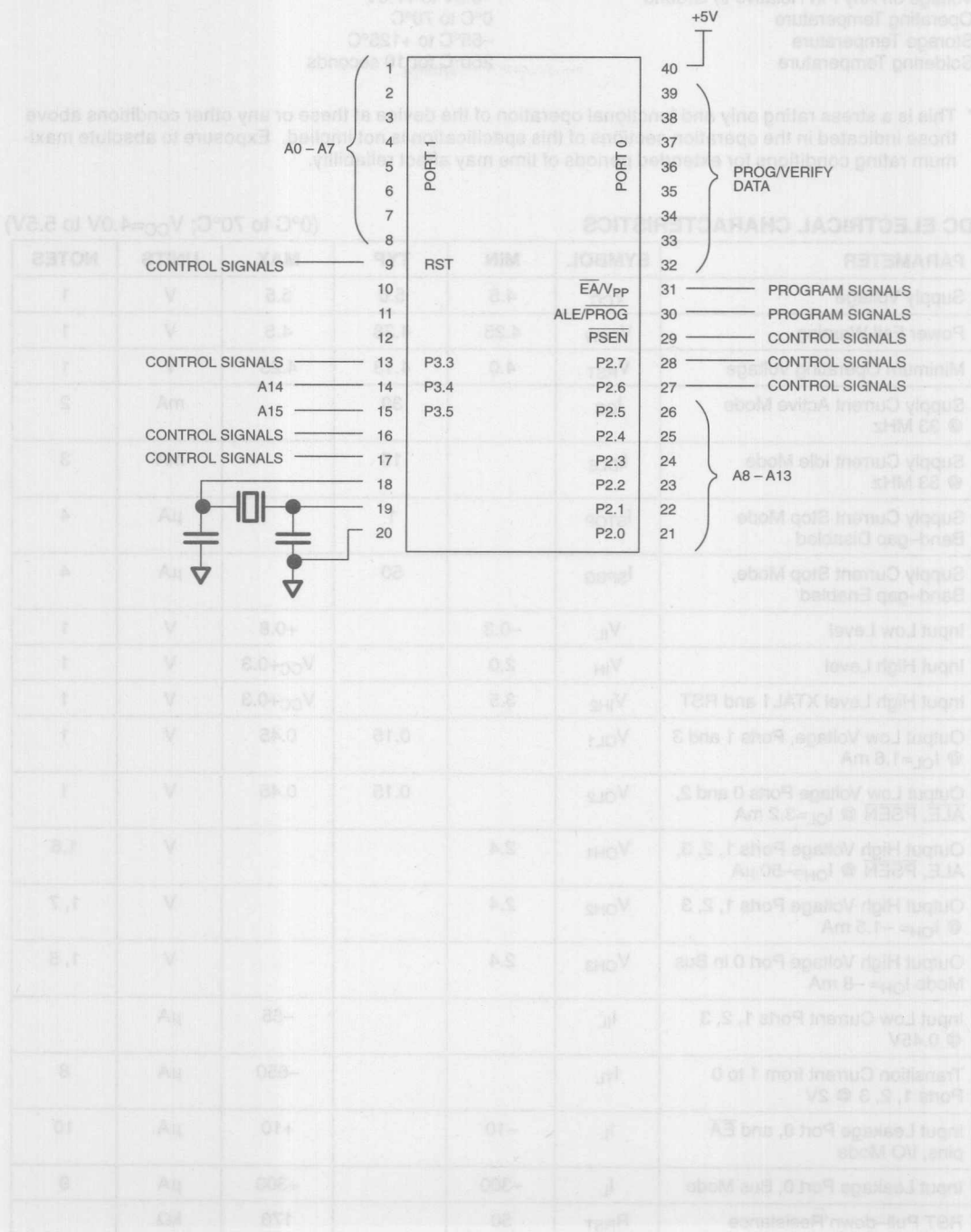
Bit 2–0 Reserved. Program to a 1.

### SIGNATURE

The Signature bytes identify the product and programming revision to EPROM programmers. This information is at programming addresses 30h, 31h, and 60h. This information is as follows::

Address	Value	Meaning
30h	DAh	Manufacturer
31h	20h	Model
60h	01h	Extension

EPROM PROGRAMMING CONFIGURATION Figure 5



**ABSOLUTE MAXIMUM RATINGS\***

Voltage on Any Pin Relative to Ground

-0.3V to +7.0V

Operating Temperature

0°C to 70°C

Storage Temperature

-55°C to +125°C

Soldering Temperature

260°C for 10 seconds

\* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

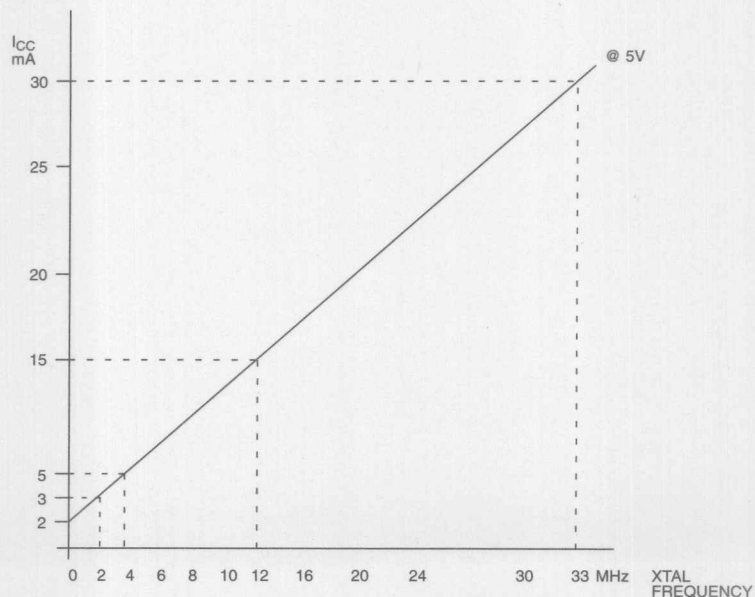
**DC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	$V_{CC}$	4.5	5.0	5.5	V	1
Power Fail Warning	$V_{PFW}$	4.25	4.38	4.5	V	1
Minimum Operating Voltage	$V_{RST}$	4.0	4.13	4.25	V	1
Supply Current Active Mode @ 33 MHz	$I_{CC}$		30		mA	2
Supply Current Idle Mode @ 33 MHz	$I_{IDLE}$		15		mA	3
Supply Current Stop Mode Band-gap Disabled	$I_{STOP}$		1		$\mu A$	4
Supply Current Stop Mode, Band-gap Enabled	$I_{SPBG}$		50		$\mu A$	4
Input Low Level	$V_{IL}$	-0.3		+0.8	V	1
Input High Level	$V_{IH}$	2.0		$V_{CC}+0.3$	V	1
Input High Level XTAL1 and RST	$V_{IH2}$	3.5		$V_{CC}+0.3$	V	1
Output Low Voltage, Ports 1 and 3 @ $I_{OL}=1.6$ mA	$V_{OL1}$		0.15	0.45	V	1
Output Low Voltage Ports 0 and 2, ALE, PSEN @ $I_{OL}=3.2$ mA	$V_{OL2}$		0.15	0.45	V	1
Output High Voltage Ports 1, 2, 3, ALE, PSEN @ $I_{OH}=-50$ $\mu A$	$V_{OH1}$	2.4			V	1,6
Output High Voltage Ports 1, 2, 3 @ $I_{OH}=-1.5$ mA	$V_{OH2}$	2.4			V	1,7
Output High Voltage Port 0 in Bus Mode $I_{OH}=-8$ mA	$V_{OH3}$	2.4			V	1,5
Input Low Current Ports 1, 2, 3 @ 0.45V	$I_{IL}$			-55	$\mu A$	
Transition Current from 1 to 0 Ports 1, 2, 3 @ 2V	$I_{TL}$			-650	$\mu A$	8
Input Leakage Port 0, and $\overline{EA}$ pins, I/O Mode	$I_L$	-10		+10	$\mu A$	10
Input Leakage Port 0, Bus Mode	$I_L$	-300		+300	$\mu A$	9
RST Pull-down Resistance	$R_{RST}$	50		170	k $\Omega$	

**NOTES FOR DC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature operation unless otherwise noted.

1. All voltages are referenced to ground.
2. Active current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=RST=5.5V$ , all other pins disconnected.
3. Idle mode current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=5.5V$ , RST at ground, all other pins disconnected.
4. Stop mode current measured with XTAL1 and RST grounded,  $V_{CC}=5.5V$ , all other pins disconnected. This value is not guaranteed. Users that are sensitive to this specification should contact Dallas Semiconductor for more information.
5. When addressing external memory.
6.  $RST=5.5V$ . This condition mimics operation of pins in I/O mode. Port 0 is tristated in reset and when at a logic high state during I/O mode.
7. During a 0 to 1 transition, a one-shot drives the ports hard for two clock cycles. This measurement reflects port in transition mode.
8. Ports 1, 2, and 3 source transition current when being pulled down externally. It reaches its maximum at approximately 2V.
9.  $0.45 < V_{IN} < V_{CC}$ . Not a high impedance input. This port is a weak address holding latch in Bus Mode. Peak current occurs near the input transition point of the latch, approximately 2V.
10.  $0.45 < V_{IN} < V_{CC}$ .  $RST=5.5V$ . This condition mimics operation of pins in I/O mode.

**TYPICAL  $I_{CC}$  VERSUS FREQUENCY** Figure 6



PARAMETER	SYMBOL	33 MHz		VARIABLE CLOCK		UNITS
		MIN	MAX	MIN	MAX	
Oscillator Frequency	$1/t_{CLCL}$	0	33	0	33	MHz
ALE Pulse Width	$t_{LHLL}$	40		$(3t_{CLCL}/2)-5$		ns
Port 0 Address Valid to ALE Low	$t_{AVLL}$	10		$(t_{CLCL}/2)-5$		ns
Address Hold after ALE Low	$t_{LLAX1}$	10		$(t_{CLCL}/2)-5$		ns
ALE Low to Valid Instruction In	$t_{LLIV}$		56		$2.5t_{CLCL}-20$	ns
ALE Low to $\overline{PSEN}$ Low	$t_{LLPL}$	10		$(t_{CLCL}/2)-5$		ns
$\overline{PSEN}$ Pulse Width	$t_{PLPH}$	55		$2t_{CLCL} - 5$		ns
$\overline{PSEN}$ Low to Valid Instr. In	$t_{PLIV}$		41		$2t_{CLCL}-20$	ns
Input Instruction Hold after $\overline{PSEN}$	$t_{PXIX}$	0		0		ns
Input Instruction Float after $\overline{PSEN}$	$t_{PXIZ}$		26		$t_{CLCL}-5$	ns
Port 0 Address to Valid Instr. In	$t_{AVIV}$		71		$3t_{CLCL}-20$	ns
Port 2 Address to Valid Instr. In	$t_{AVIV2}$		81		$3.5t_{CLCL}-25$	ns
$\overline{PSEN}$ Low to Address Float	$t_{PLAZ}$		0		0	ns

#### NOTES FOR AC ELECTRICAL CHARACTERISTICS:

All parameters apply to both commercial and industrial temperature range operation unless otherwise noted. All signals characterized with load capacitance of 80 pF except Port 0, ALE,  $\overline{PSEN}$ ,  $\overline{RD}$  and  $\overline{WR}$  with 100 pF. Interfacing to memory devices with float times (turn off times) over 25 ns may cause contention. This will not damage the parts, but will cause an increase in operating current.

**MOVX CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	VARIABLE CLOCK		UNITS	STRETCH
		MIN	MAX		
Data Access ALE Pulse Width	$t_{LHLL2}$	$1.5t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Address Hold after ALE Low for MOVX Write	$t_{LLAX2}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Pulse Width	$t_{RLRH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{WR}$ Pulse Width	$t_{WLWH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Valid Data In	$t_{RLDV}$		$2t_{CLCL}-20$ $t_{MCS}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold after Read	$t_{RHDX}$	0		ns	
Data Float after Read	$t_{RHDZ}$		$t_{CLCL}-5$ $2t_{CLCL}-5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to Valid Data In	$t_{LLDV}$		$2.5t_{CLCL}-20$ $t_{MCS} + t_{CLCL}-40$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to Valid Data In	$t_{AVDV1}$		$3t_{CLCL}-20$ $t_{MCS}+1.5t_{CLCL}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to Valid Data In	$t_{AVDV2}$		$3.5t_{CLCL}-20$ $t_{MCS}+2t_{CLCL}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$t_{LLWL}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$	$0.5t_{CLCL}+5$ $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL1}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL2}$	$1.5t_{CLCL}-10$ $2.5t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Valid to $\overline{WR}$ Transition	$t_{QVWX}$	-5		ns	
Data Hold after Write	$t_{WHQX}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Address Float	$t_{RLAZ}$		$-0.5t_{CLCL}-5$	ns	
$\overline{RD}$ or $\overline{WR}$ High to ALE High	$t_{WHLH}$	0 $t_{CLCL}-5$	10 $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$

NOTE:  $t_{MCS}$  is a time period related to the Stretch memory cycle selection. The following table shows the value of  $t_{MCS}$  for each Stretch selection.

M2	M1	M0	MOVX CYCLES	$t_{MCS}$
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	$4 t_{CLCL}$
0	1	0	4 machine cycles	$8 t_{CLCL}$
0	1	1	5 machine cycles	$12 t_{CLCL}$
1	0	0	6 machine cycles	$16 t_{CLCL}$
1	0	1	7 machine cycles	$20 t_{CLCL}$
1	1	0	8 machine cycles	$24 t_{CLCL}$
1	1	1	9 machine cycles	$28 t_{CLCL}$

**EXTERNAL CLOCK CHARACTERISTICS**(0°C to 70°C; V<sub>CC</sub>=4.0V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Clock High Time	t <sub>CHCX</sub>	10			ns	
Clock Low Time	t <sub>CLCX</sub>	10			ns	
Clock Rise Time	t <sub>CLCL</sub>			5	ns	
Clock Fall Time	t <sub>CHCL</sub>			5	ns	

**SERIAL PORT MODE 0 TIMING CHARACTERISTICS**(0°C to 70°C; V<sub>CC</sub>=4.0V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Serial Port Clock Cycle Time SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	t <sub>XLXL</sub>		12t <sub>CLCL</sub> 4t <sub>CLCL</sub>		ns ns	
Output Data Setup to Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	t <sub>QVXH</sub>		10t <sub>CLCL</sub> 3t <sub>CLCL</sub>		ns ns	
Output Data Hold from Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	t <sub>XHQX</sub>		2t <sub>CLCL</sub> t <sub>CLCL</sub>		ns ns	
Input Data Hold after Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	t <sub>XHDX</sub>		t <sub>CLCL</sub> t <sub>CLCL</sub>		ns ns	
Clock Rising Edge to Input Data Valid SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	t <sub>XHDV</sub>		11t <sub>CLCL</sub> 3t <sub>CLCL</sub>		ns ns	

**EXPLANATION OF AC SYMBOLS**

In an effort to remain compatible with the original 8051 family, this device specifies the same parameters as such devices, using the same symbols. For completeness, the following is an explanation of the symbols.

t	Time			
A	Address			
C	Clock			
D	Input data			
H	Logic level high			
L	Logic level low			
I	Instruction			
P	PSEN			
Q	Output data			
R	RD signal			
V	Valid			
W	WR signal			
X	No longer a valid logic level			
Z	Tristate			

**POWER CYCLE TIMING CHARACTERISTICS** (0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Cycle Start-up Time	$t_{CSU}$		1.8		ms	1
Power-on Reset Delay	$t_{POR}$			65536	$t_{CLCL}$	2

**NOTES FOR POWER CYCLE TIMING CHARACTERISTICS:**

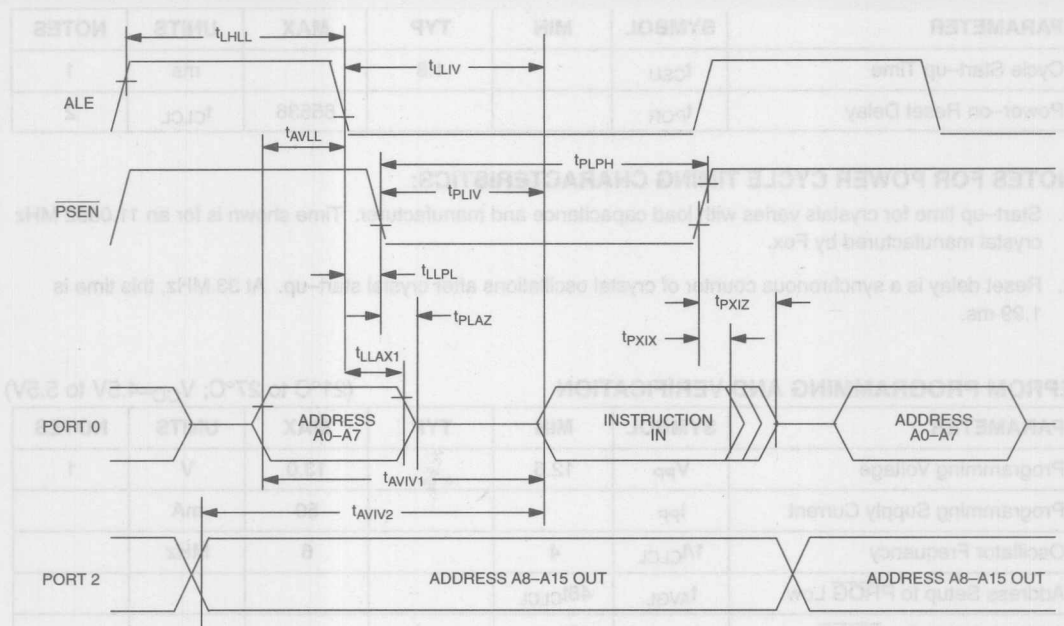
1. Start-up time for crystals varies with load capacitance and manufacturer. Time shown is for an 11.0592 MHz crystal manufactured by Fox.
2. Reset delay is a synchronous counter of crystal oscillations after crystal start-up. At 33 MHz, this time is 1.99 ms.

**EPROM PROGRAMMING AND VERIFICATION** (21°C to 27°C;  $V_{CC}=4.5V$  to 5.5V)

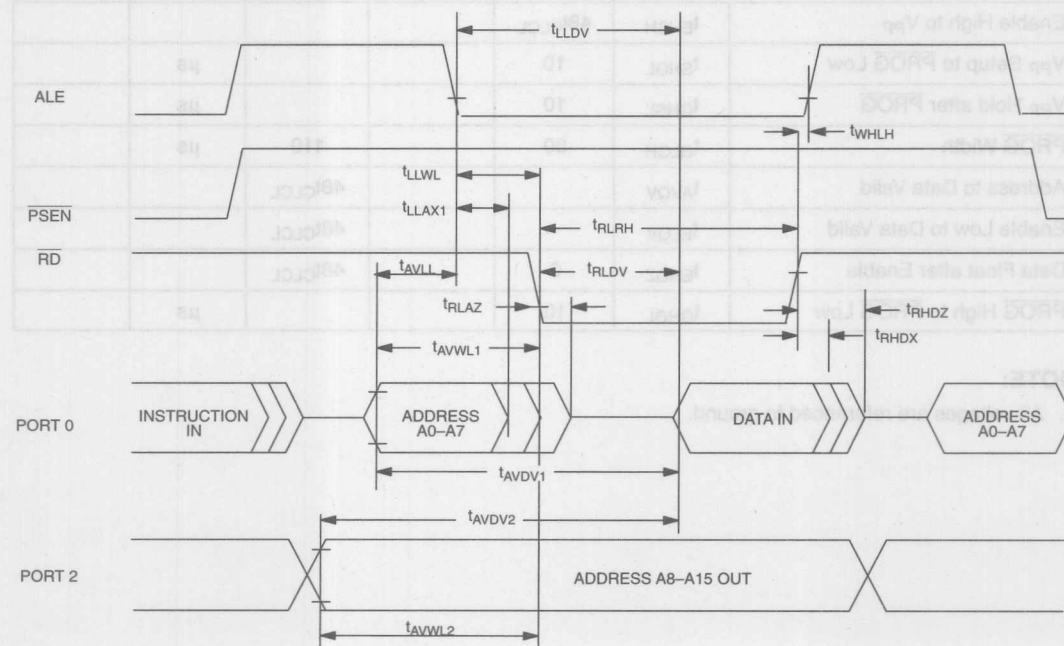
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Programming Voltage	$V_{PP}$	12.5		13.0	V	1
Programming Supply Current	$I_{PP}$			50	mA	
Oscillator Frequency	$1/t_{CLCL}$	4		6	MHz	
Address Setup to $\overline{PROG}$ Low	$t_{AVGL}$	$48t_{CLCL}$				
Address Hold after $\overline{PROG}$	$t_{GHAX}$	$48t_{CLCL}$				
Data Setup to $\overline{PROG}$ Low	$t_{DVGL}$	$48t_{CLCL}$				
Data Hold after $\overline{PROG}$	$t_{GHDX}$	$48t_{CLCL}$				
Enable High to $V_{PP}$	$t_{EHS}$	$48t_{CLCL}$				
$V_{PP}$ Setup to $\overline{PROG}$ Low	$t_{SHGL}$	10			$\mu s$	
$V_{PP}$ Hold after $\overline{PROG}$	$t_{GHSL}$	10			$\mu s$	
$\overline{PROG}$ Width	$t_{GLGH}$	90		110	$\mu s$	
Address to Data Valid	$t_{AVQV}$			$48t_{CLCL}$		
Enable Low to Data Valid	$t_{ELQV}$			$48t_{CLCL}$		
Data Float after Enable	$t_{EHQZ}$	0		$48t_{CLCL}$		
$\overline{PROG}$ High to $\overline{PROG}$ Low	$t_{GHGL}$	10			$\mu s$	

**NOTE:**

1. All voltages are referenced to ground.

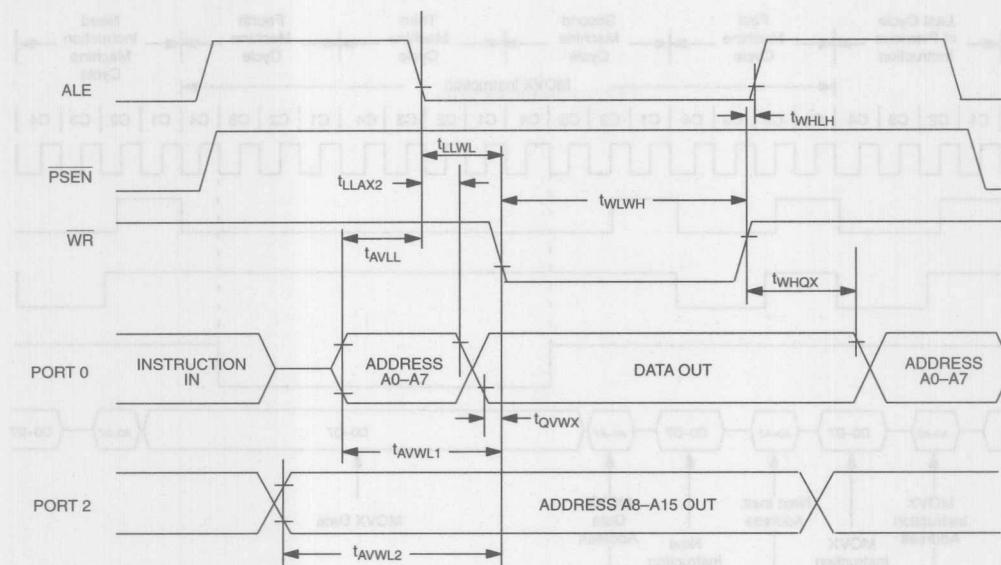


## EXTERNAL DATA MEMORY READ CYCLE

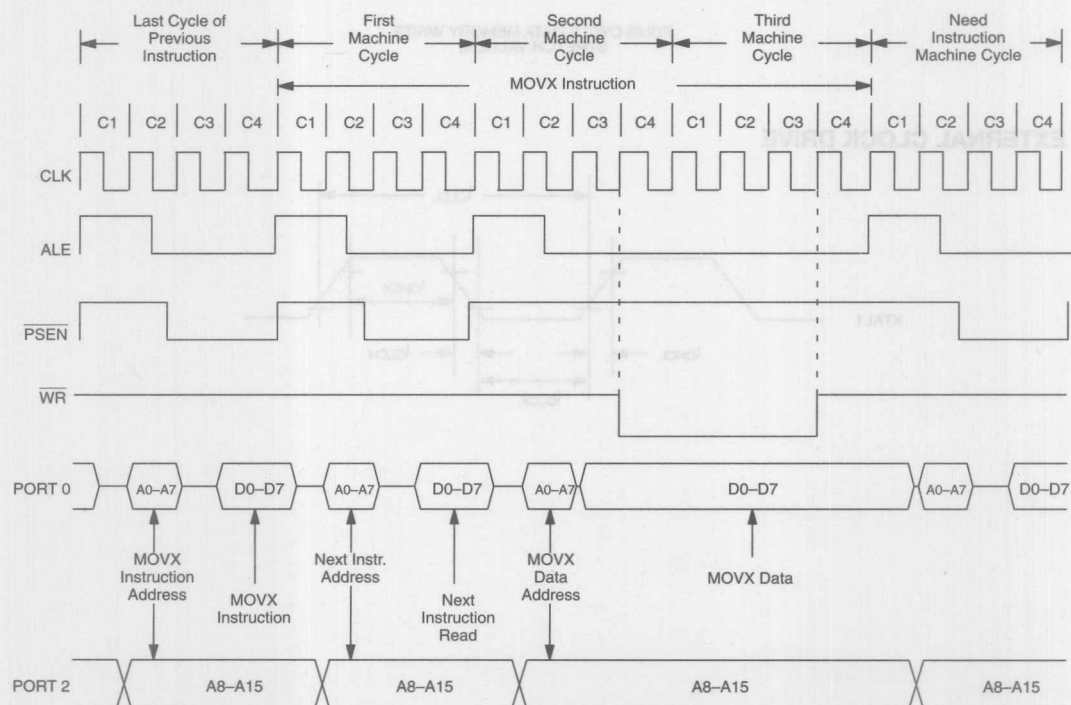




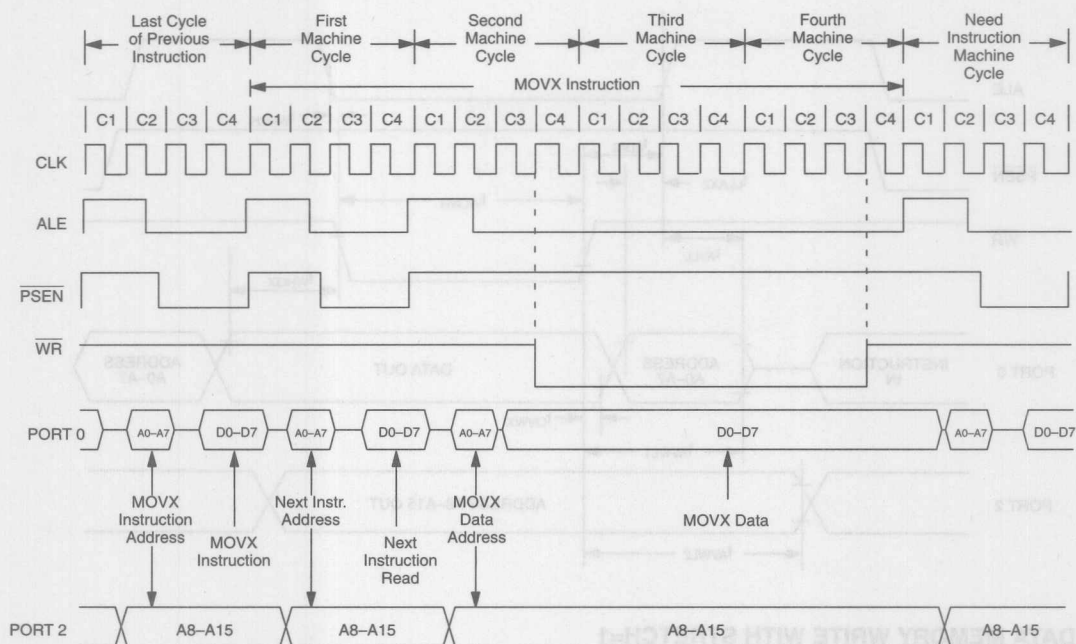
## EXTERNAL DATA MEMORY WRITE CYCLE



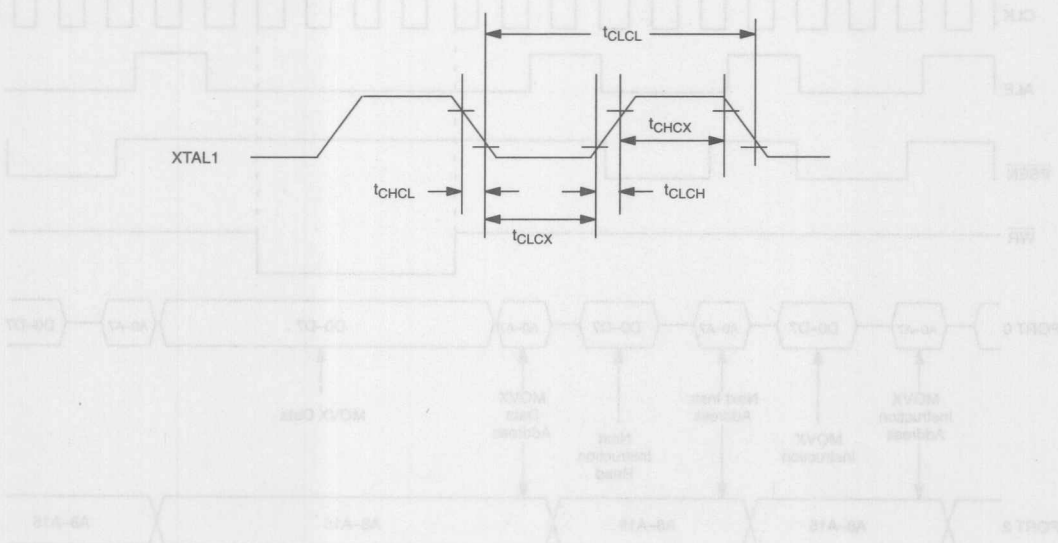
## DATA MEMORY WRITE WITH STRETCH=1



## DATA MEMORY WRITE WITH STRETCH=2

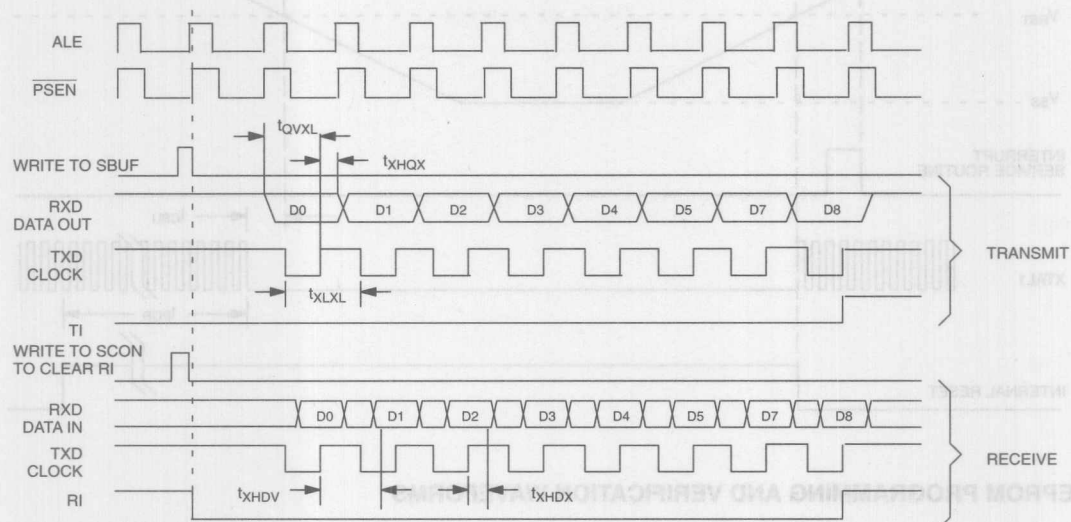
FOUR CYCLE DATA MEMORY WRITE  
STRETCH VALUE=2

## EXTERNAL CLOCK DRIVE

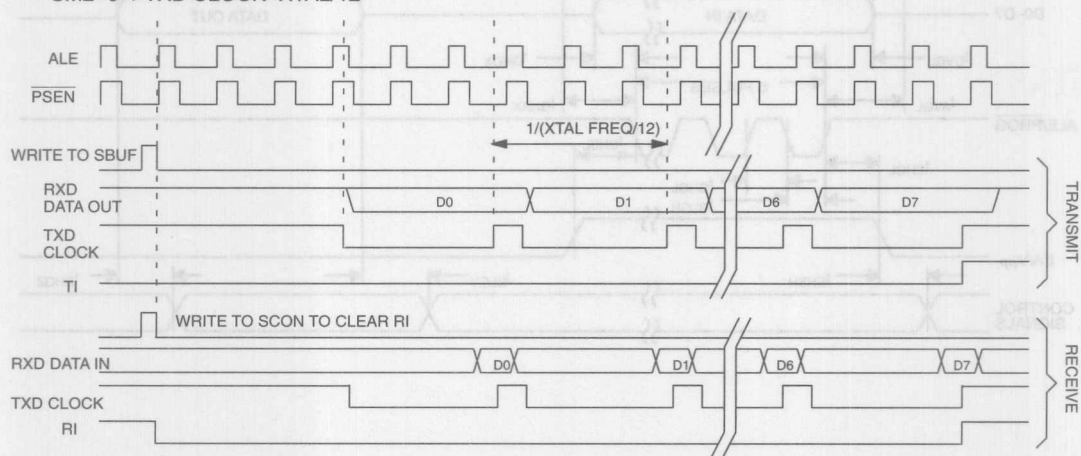


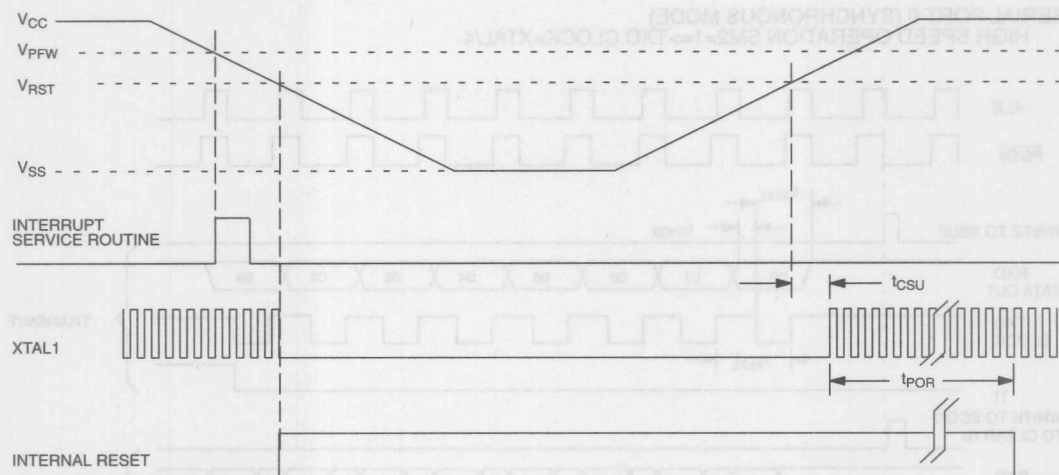
# SERIAL PORT MODE 0 TIMING

SERIAL PORT 0 (SYNCHRONOUS MODE)  
HIGH SPEED OPERATION SM2=1=>TXD CLOCK=XTAL/4

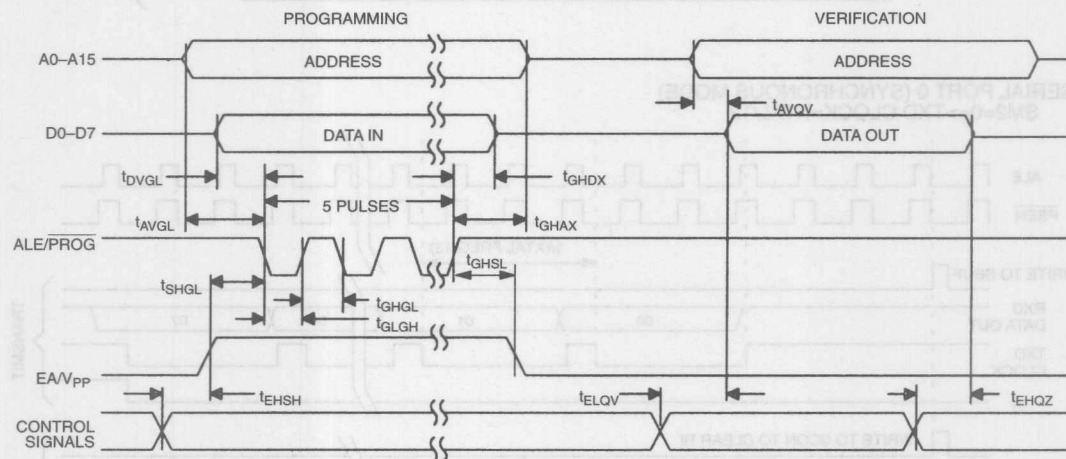


SERIAL PORT 0 (SYNCHRONOUS MODE)  
SM2=0=>TXD CLOCK=XTAL/12

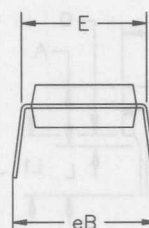
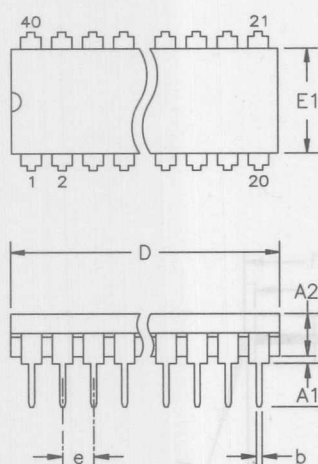




## EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



## 40-PIN PDIP (600 MIL)



ALL DIMENSIONS ARE IN INCHES.

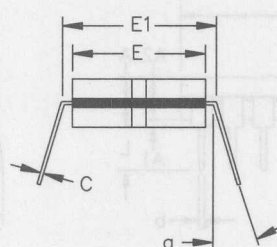
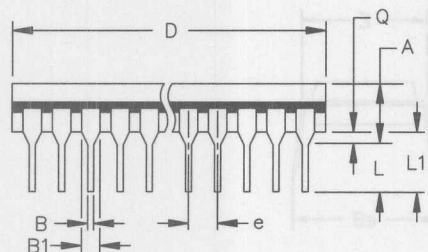
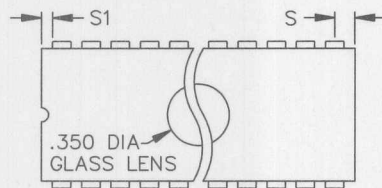
PKG	40-PIN	
DIM	MIN	MAX
A	—	0.200
A1	0.015	—
A2	0.140	0.160
b	0.014	0.022
c	0.008	0.012
D	1.980	2.085
E	0.600	0.625
E1	0.530	0.555
e	0.090	0.110
L	0.115	0.145
eB	0.600	0.700

56-G5000-000

PKG	40-PIN	
DIM	MIN	MAX
A	—	0.200
A1	0.015	—
A2	0.140	0.160
b	0.014	0.022
c	0.008	0.012
D	1.980	2.085
E	0.600	0.625
E1	0.530	0.555
e	0.090	0.110
L	0.115	0.145
eB	0.600	0.700



## 40-PIN CER DIP



ALL DIMENSIONS ARE IN INCHES.

ALL DIMENSIONS ARE IN INCHES.

PKG	40-PIN	
DIM	MIN	MAX
A	—	0.225
B	0.014	0.023
B1	0.038	0.065
C	0.006	0.015
D	—	2.096
E	0.510	0.620
E1	0.590	0.630
e	100 BSC	
L	0.125	0.200
L1	0.150	—
Q	0.020	0.060
S	—	0.098
S1	0.005	—
a	0°	15°

56-G4008-001

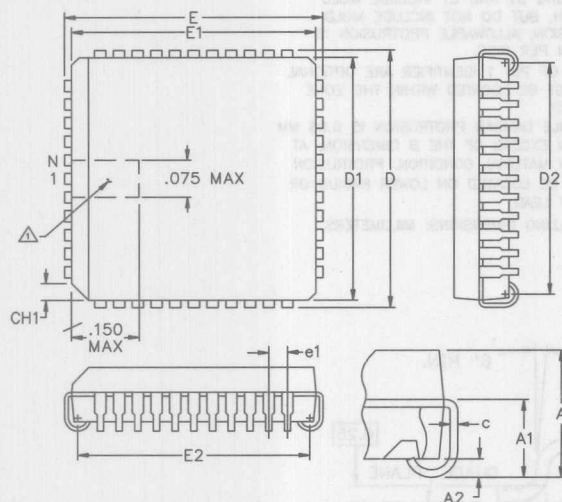
40-PIN	40-PIN	40-PIN
MAX	MIN	MIN
0.005	—	A
—	0.014	B
0.038	0.014	B1
0.006	0.014	C
0.005	0.005	e
0.005	0.005	D
0.005	0.005	E
0.005	0.005	E1
0.005	0.005	a
0.005	0.005	J
0.005	0.005	Se

56-G4008-001

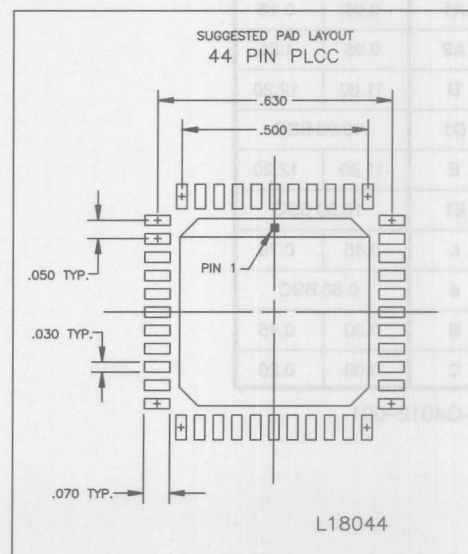
## 44-PIN PLCC

NOTE:

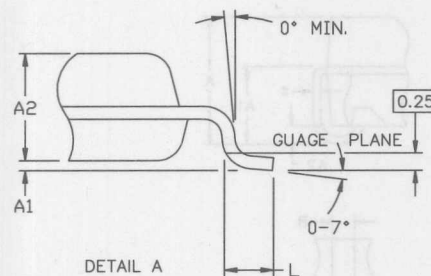
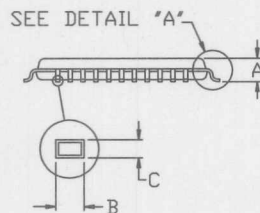
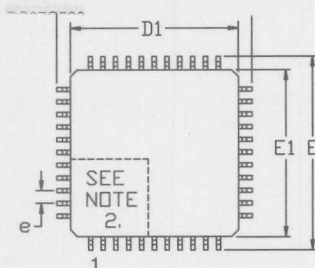
1.  $\triangle$  PIN-1 IDENTIFIER TO BE LOCATED IN ZONE INDICATED.
2. CONTROLLING DIMENSIONS ARE IN INCHS



PKG	44-PIN	
DIM	MIN	MAX
A	0.165	0.180
A1	0.090	0.120
A2	0.020	—
B	0.026	0.033
B1	0.013	0.021
c	0.009	0.012
CH1	0.042	0.048
D	0.685	0.695
D1	0.650	0.656
D2	0.590	0.630
E	0.685	0.695
E1	0.650	0.656
E2	0.590	0.630
e1	0.050 BSC	
N	44	—



56-G4003-001



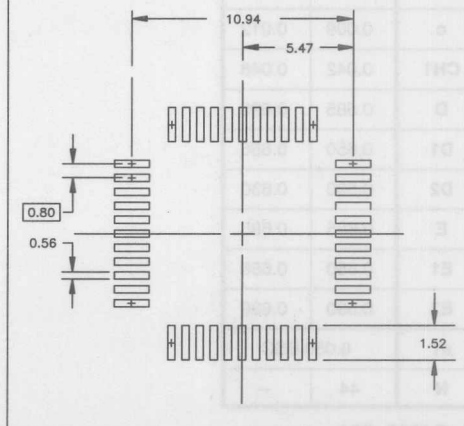
PKG	44-PIN	
DIM	MIN	MAX
A	-	1.20
A1	0.05	0.15
A2	0.95	1.05
D	11.80	12.20
D1	10.00 BSC	
E	11.80	12.20
E1	10.00 BSC	
L	0.45	0.75
e	0.80 BSC	
B	0.30	0.45
C	0.09	0.20

56-G4012-001

#### NOTES:

1. DIMENSIONS D1 AND E1 INCLUDE MOLD MISMATCH, BUT DO NOT INCLUDE MOLD PROTRUSION; ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE.
2. DETAILS OF PIN 1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATED.
3. ALLOWABLE DAMBAR PROTRUSION IS 0.08 MM TOTAL IN EXCESS OF THE B DIMENSION; AT MAXIMUM MATERIAL CONDITION. PROTRUSION NOT TO BE LOCATED ON LOWER RADIUS OR FOOT OF LEAD.
4. CONTROLLING DIMENSIONS: MILLIMETERS.

#### SUGGESTED PAD LAYOUT 44 PIN TQFP, 10\*10\*1.0



**DALLAS**  
 SEMICONDUCTOR

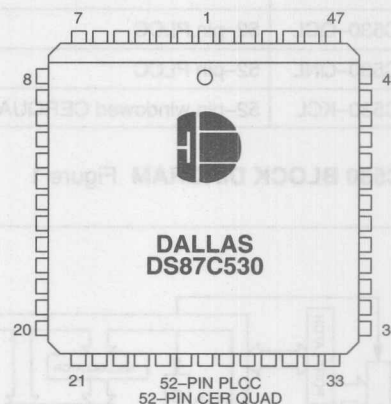
# DS87C530

## EPROM Micro with Real Time Clock

### FEATURES

- 80C52 Compatible
  - 8051 Instruction set
  - Four 8-bit I/O ports
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
- Large On-chip Memory
  - 16KB EPROM (OTP)
  - 1KB extra on-chip SRAM for MOVX
- **ROMSIZE™ Feature**
  - Selects effective on-chip ROM size from 0 to 16KB
  - Allows access to entire external memory map
  - Dynamically adjustable by software
  - Useful as boot block for external Flash
- Nonvolatile Functions
  - **On-chip Real-time clock w/ Alarm Interrupt**
  - **Battery backup support of 1KB SRAM**
- High-Speed Architecture
  - 4 clocks/machine cycle ( $8051 = 12$ )
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM/peripherals
- Power Management Mode
  - Programmable clock source saves power
  - Runs from (crystal/64) or (crystal/1024)
  - Provides automatic hardware and software exit
- EMI Reduction Mode disables ALE
- High integration controller includes:
  - Power-fail reset
  - Early-warning power-fail interrupt
  - Programmable Watchdog timer
- Two full-duplex hardware serial ports
- 14 total interrupt sources with 6 external

### PACKAGE OUTLINE



### DESCRIPTION

The DS87C530 is an 8051 compatible microcontroller based on the Dallas High Speed core. It uses four clocks per instruction cycle instead of 12 used by the standard 8051. It also provides a unique mix of peripherals not widely available on other processors. They include an on-chip real-time clock (RTC) and battery back up support for an on-chip 1K x 8 SRAM. The new Power Management Mode allows software to select reduced power operation while still processing.

A combination of high performance microcontroller core, real-time clock, battery backed SRAM, and power management makes the DS87C530 ideal for instruments and portable applications. It also provides several peripherals found on other Dallas High-Speed Microcontrollers. These include two independent serial ports, two data pointers, on-chip power monitor with brown-out detection and a watchdog timer.

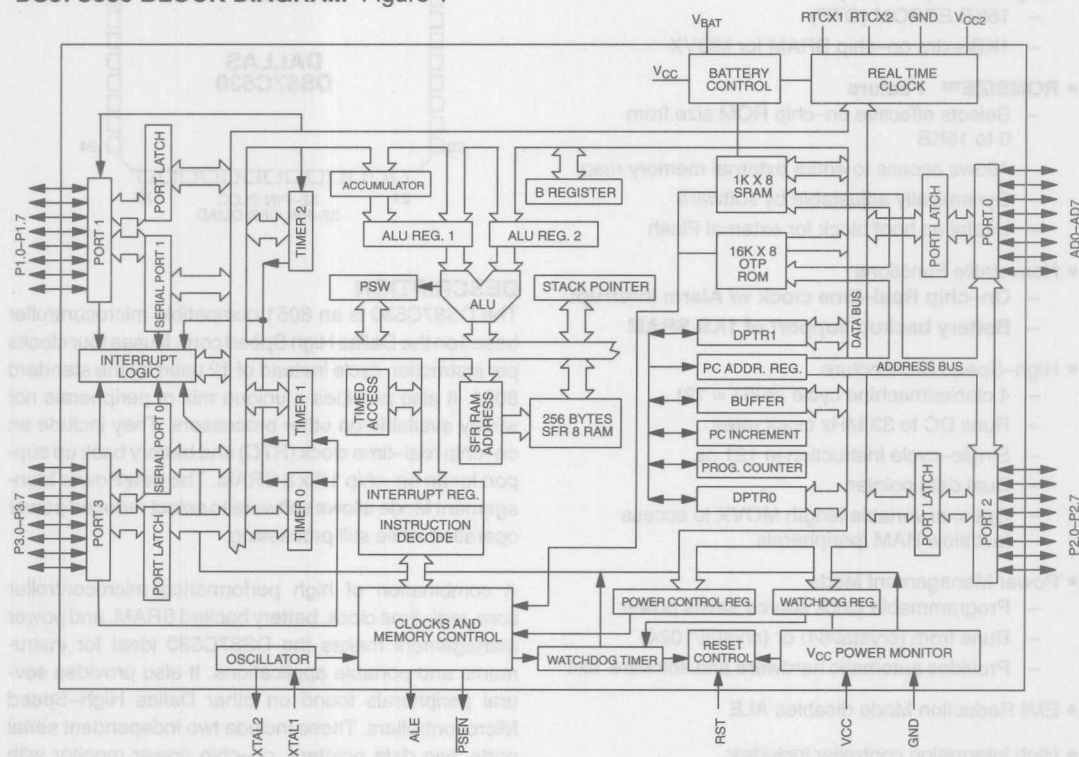
Power Management Mode (PMM) allows software to select a slower CPU clock. While default operation uses four clocks per machine cycle, the PMM runs the processor at 64 or 1024 clocks per cycle. There is a corresponding drop in power consumption when the processor slows.

Note: The DS87C530 is a monolithic device. A user must supply an external battery or super-cap and a 32.768 KHz timekeeping crystal to have permanently powered timekeeping or nonvolatile RAM. The DS87C530 provides all the support and switching circuitry needed to manage these resources.

## ORDERING INFORMATION

PART NUMBER	PACKAGE	MAX. CLOCK SPEED	TEMPERATURE RANGE
DS87C530-QCL	52-pin PLCC	33 MHz	0°C to 70°C
DS87C530-QNL	52-pin PLCC	33 MHz	-40°C to +85°C
DS87C530-KCL	52-pin windowed CERQUAD	33 MHz	0°C to 70°C

DS87C530 BLOCK DIAGRAM Figure 1





PIN DESCRIPTION Table 1

PLCC	SIGNAL NAME	DESCRIPTION
52	V <sub>CC</sub>	V <sub>CC</sub> – +5V. Processor power supply.
1,25	GND	GND – Processor digital circuit ground.
29	V <sub>CC2</sub>	V <sub>CC2</sub> – +5V Real-time clock supply.
26	GND2	GND2 – Real-time clock circuit ground.
12	RST	<b>RST – Input.</b> The RST input pin contains a Schmitt voltage input to recognize external active high Reset inputs. The pin also employs an internal pull-down resistor to allow for a combination of wired OR external Reset sources. An RC is not required for power-up, as the DS87C530 provides this function internally.
23 24	XTAL2 XTAL1	<b>XTAL1, XTAL2</b> – The crystal oscillator pins XTAL1 and XTAL2 provide support for parallel resonant, AT cut crystals. XTAL1 acts also as an input if there is an external clock source in place of a crystal. XTAL2 serves as the output of the crystal amplifier.
38	PSEN	<b>PSEN – Output.</b> The Program Store Enable output. This signal is commonly connected to optional external ROM memory as a chip enable. PSEN will provide an active low pulse and is driven high when external ROM is not being accessed.
39	ALE	<b>ALE – Output.</b> The Address Latch Enable output functions as a clock to latch the external address LSB from the multiplexed address/data bus on Port 0. This signal is commonly connected to the latch enable of an external 373 family transparent latch. ALE has a pulse width of 1.5 XTAL1 cycles and a period of four XTAL1 cycles. ALE is forced high when the DS87C530 is in a Reset condition. ALE can also be disabled using the EMI reduction mode.
50 49 48 47 46 45 44 43	P0.0 (AD0) P0.1 (AD1) P0.2 (AD2) P0.3 (AD3) P0.4 (AD4) P0.5 (AD5) P0.6 (AD6) P0.7 (AD7)	<b>Port 0 (AD0–7) – I/O.</b> Port 0 is an <u>open-drain</u> 8-bit bi-directional I/O port. As an alternate function Port 0 can function as the multiplexed address/data bus to access off-chip memory. During the time when ALE is high, the LSB of a memory address is presented. When ALE falls to a logic 0, the port transitions to a bi-directional data bus. This bus is used to read external ROM and read/write external RAM memory or peripherals. When used as a memory bus, the port provides active high drivers. The reset condition of Port 0 is tri-state. Pull-up resistors are required when using Port 0 as an I/O port.
3–10	P1.0 – P1.7	<b>Port 1 – I/O.</b> Port 1 functions as both an 8-bit bi-directional I/O port and an alternate functional interface for Timer 2 I/O, new External Interrupts, and new Serial Port 1. The reset condition of Port 1 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C530 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes the output high (and input) state. The alternate modes of Port 1 are outlined as follows.

3		P1.0	T2	External I/O for Timer/Counter 2
4		P1.1	T2EX	Timer/Counter 2 Capture/Reload Trigger
5		P1.2	RXD1	Serial Port 1 Input
6		P1.3	TXD1	Serial Port 1 Output
7		P1.4	INT2	External Interrupt 2 (Positive Edge Detect)
8		P1.5	INT3	External Interrupt 3 (Negative Edge Detect)
9		P1.6	INT4	External Interrupt 4 (Positive Edge Detect)
10		P1.7	INT5	External Interrupt 5 (Negative Edge Detect)
30	P2.0 (AD8)	<b>Port 2 (A8–15) – I/O.</b> Port 2 is a bi-directional I/O port. The reset condition of Port 2 is logic high. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C530 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes both the output high and input state. As an alternate function Port 2 can function as MSB of the external address bus. This bus can be used to read external ROM and read/write external RAM memory or peripherals.		
31	P2.1 (AD9)			
32	P2.2 (AD10)			
33	P2.3 (AD11)			
34	P2.4 (AD12)			
35	P2.5 (AD13)			
36	P2.6 (AD14)			
37	P2.7 (AD15)			
15–22	P3.0 – P3.7	<b>Port 3 – I/O.</b> Port 3 functions as both an 8-bit bi-directional I/O port and an alternate functional interface for External Interrupts, Serial Port 0, Timer 0 and 1 Inputs, and $\overline{RD}$ and $\overline{WR}$ strobes. The reset condition of Port 3 is with all bits at a logic 1. In this state, a weak pull-up holds the port high. This condition also serves as an input mode, since any external circuit that writes to the port will overcome the weak pull-up. When software writes a 0 to any port pin, the DS87C530 will activate a strong pull-down that remains on until either a 1 is written or a reset occurs. Writing a 1 after the port has been at 0 will cause a strong transition driver to turn on, followed by a weaker sustaining pull-up. Once the momentary strong driver turns off, the port again becomes both the output high and input state. The alternate modes of Port 3 are outlined below.		
		<b>Port</b>	<b>Alternate Mode</b>	
15		P3.0	RXD0	Serial Port 0 Input
16		P3.1	TXD0	Serial Port 0 Output
17		P3.2	INT0	External Interrupt 0
18		P3.3	INT1	External Interrupt 1
19		P3.4	T0	Timer 0 External Input
20		P3.5	T1	Timer 1 External Input
21		P3.6	$\overline{WR}$	External Data Memory Write Strobe
22		P3.7	$\overline{RD}$	External Data Memory Read Strobe
42	$\overline{EA}$	<b><math>\overline{EA}</math> – Input.</b> Connect to ground to force the DS87C530 to use an external ROM. The internal RAM is still accessible as determined by register settings. Connect $\overline{EA}$ to $V_{CC}$ to use internal ROM.		
51	$V_{BAT}$	<b><math>V_{BAT}</math> – Input.</b> Connect to the power source that maintains SRAM and RTC when $V_{CC} < V_{BAT}$ . May be connected to a 3V lithium battery or a super-cap. See the electrical specifications for details.		

PLCC	SIGNAL NAME	DESCRIPTION
27, 28	RTCX2, RTCX1	<b>RTCX2, RTCX1 – Timekeeping crystal.</b> Connect a 32.768 KHz crystal between RTCX2 and RTCX1 to supply the time-base for the real-time clock. The DS87C530 supports both 6 pF and 12.5 pF load capacitance crystals as selected by an SFR bit described below. To prevent noise from affecting the RTC, the RTCX2 and RTCX1 pin should be guard-ringed with GND2.
2, 11, 13, 14, 40, 41	NC	<b>NC – Reserved.</b> These pins should not be connected. They are reserved for use with future devices in the family.

### COMPATIBILITY

The DS87C530 is a fully static CMOS 8051 compatible microcontroller designed for high performance. While remaining familiar to 8051 users, it has many new features. In general, software written for existing 8051 based systems works without modification on the DS87C530. The exception is critical timing since the High Speed Micro performs its instructions much faster than the original for any given crystal selection. The DS87C530 runs the standard 8051 instruction set. It is not pin compatible with other 8051s due to the time-keeping crystal.

The DS87C530 provides three 16-bit timer/counters, full-duplex serial port (2), 256 bytes of direct RAM plus 1KB of extra MOVX RAM. I/O ports have the same operation as a standard 8051 product. Timers will default to a 12 clock per cycle operation to keep their timing compatible with original 8051 systems. However, timers are individually programmable to run at the new 4 clocks per cycle if desired. The PCA is not supported.

The DS87C530 provides several new hardware features implemented by new Special Function Registers. A summary of these SFRs is provided below.

### PERFORMANCE OVERVIEW

The DS87C530 features a high speed 8051 compatible core. Higher speed comes not just from increasing the clock frequency, but from a newer, more efficient design.

This updated core does not have the dummy memory cycles that are present in a standard 8051. A conventional 8051 generates machine cycles using the clock frequency divided by 12. In the DS87C530, the same machine cycle takes four clocks. Thus the fastest instruction, 1 machine cycle, executes three times faster for the same crystal frequency. Note that these are identical instructions. The majority of instructions on

the DS87C530 will see the full 3 to 1 speed improvement. Some instructions will get between 1.5 and 2.4 to 1 improvement. All instructions are faster than the original 8051.

The numerical average of all opcodes gives approximately a 2.5 to 1 speed improvement. Improvement of individual programs will depend on the actual instructions used. Speed sensitive applications would make the most use of instructions that are three times faster. However, the sheer number of 3 to 1 improved opcodes makes dramatic speed improvements likely for any code. These architecture improvements and 0.8  $\mu$ m CMOS produce a peak instruction cycle in 121 ns (8.25 MIPs). The Dual Data Pointer feature also allows the user to eliminate wasted instructions when moving blocks of memory.

### INSTRUCTION SET SUMMARY

All instructions in the DS87C530 perform the same functions as their 8051 counterparts. Their effect on bits, flags, and other status functions is identical. However, the timing of each instruction is different. This applies both in absolute and relative number of clocks.

For absolute timing of real-time events, the timing of software loops can be calculated using a table in the High-Speed Microcontroller User's Guide. However, counter/timers default to run at the older 12 clocks per increment. In this way, timer-based events occur at the standard intervals with software executing at higher speed. Timers optionally can run at 4 clocks per increment to take advantage of faster processor operation.

The relative time of two instructions might be different in the new architecture than it was previously. For example, in the original architecture, the "MOVX A, @DPTR" instruction and the "MOV direct, direct" instruction used two machine cycles or 24 oscillator cycles. Therefore, they required the same amount of time. In the

DS87C530, the MOVX instruction takes as little as two machine cycles or eight oscillator cycles but the "MOV direct, direct" uses three machine cycles or 12 oscillator cycles. While both are faster than their original counterparts, they now have different execution times. This is because the DS87C530 usually uses one instruction cycle for each instruction byte. The user concerned with precise program timing should examine the timing of

each instruction for familiarity with the changes. Note that a machine cycle now requires just four clocks, and provides one ALE pulse per cycle. Many instructions require only one cycle, but some require five. In the original architecture, all were one or two cycles except for MUL and DIV. Refer to the High-Speed Microcontroller User's Guide for details and individual instruction timing.

## SPECIAL FUNCTION REGISTER LOCATIONS Table 2

\* Functions not present in the 80C52 are in bold

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
SP									81h
DPL									82h
DPH									83h
<b>DPL1</b>									84h
<b>DPH1</b>									85h
<b>DPS</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>SEL</b>	86h
PCON	SMOD_0	<b>SMOD0</b>	—	—	GF1	GF0	STOP	IDLE	87h
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89h
TL0									8Ah
TL1									8Bh
TH0									8Ch
TH1									8Dh
<b>CKCON</b>	<b>WD1</b>	<b>WD0</b>	<b>T2M</b>	<b>T1M</b>	<b>T0M</b>	<b>MD2</b>	<b>MD1</b>	<b>MD0</b>	8Eh
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
<b>EXIF</b>	<b>IE5</b>	<b>IE4</b>	<b>IE3</b>	<b>IE2</b>	<b>XT/RG</b>	<b>RGMD</b>	<b>RGSL</b>	<b>BGS</b>	91h
<b>TRIM</b>	<b>E4K</b>	<b>X12/6</b>	<b>TRM2</b>	<b>TRM2</b>	<b>TRM1</b>	<b>TRM1</b>	<b>TRM0</b>	<b>TRM0</b>	96h
SCON0	SM0/FE_0	SM1_0	SM2_0	REN_0	TB8_0	RB8_0	TI_0	RI_0	98h
SBUF0									99h
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
IE	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	A8h
SADDR0									A9h
SADDR1									AAh
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
IP	—	PS1	PT2	PS0	PT1	PX1	PT0	PX0	B8h
SADEN0									B9h

REGISTER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	ADDRESS
SADEN1									BAh
SCON1	SM0/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TL_1	RI_1	C0h
SBUF1									C1h
ROMSIZE	—	—	—	—	—	RMS2	RMS1	RMS0	C2h
PMR	CD1	CD0	SWB	—	XTOFF	ALEOFF	DME1	DME0	C4h
STATUS	PIP	HIP	LIP	XTUP	SPTA1	SPRA1	SPTA0	SPRA0	C5h
TA									C7h
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
T2MOD	—	—	—	—	—	—	T2OE	DCEN	C9h
RCAP2L									CAh
RCAP2H									CBh
TL2									CCh
TH2									CDh
PSW	CY	AC	F0	RS1	RS0	OV	FL	P	D0h
WDCON	SMOD_1	POR	EPFI	PFI	WDIF	WTRF	EWI	RWT	D8h
ACC									E0h
EIE	—	—	ERTCI	EWDI	EX5	EX4	EX3	EX2	E8h
B									F0h
RTASS									F2h
RTAS	0	0							F3h
RTAM	0	0							F4h
RTAH	0	0	0						F5h
EIP	—	—	PRTCI	PWDI	PX5	PX4	PX3	PX2	F8h
RTCC	SSCE	SCE	MCE	HCE	RTCRE	RTCWE	RTCIF	RTCE	F9h
RTCSS									FAh
RTCS	0	0							FBh
RTCM	0	0							FCh
RTCH									FDh
RTCD0									FEh
RTCD1									FFh





merely powered if a user supplies an external energy source. These are an on-chip real-time clock and a nonvolatile SRAM. The chip contains all related functions and controls. The user must supply a backup source and a 32.768 KHz timekeeping crystal.

## REAL TIME CLOCK

The on-chip real-time clock (RTC) keeps time of day and calendar functions. Its timebase is a 32.768 KHz crystal between pins RTCX1 and RTCX2. The RTC maintains time to 1/256 of a second. It also allows a user to read (and write) seconds, minutes, hours, day of the week, and date. The clock organization is shown in Figure 2.

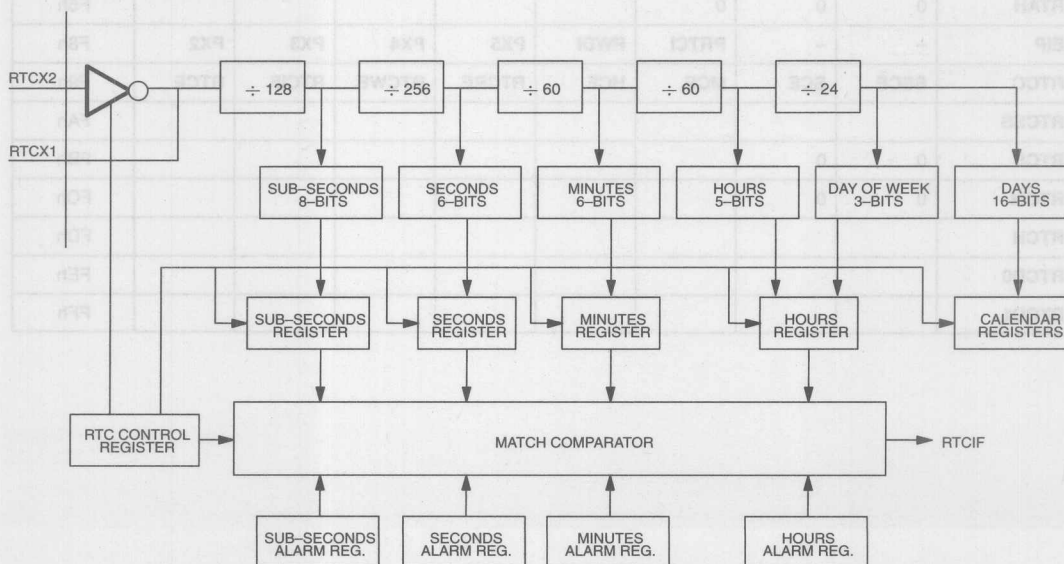
Timekeeping registers allow easy access to commonly needed time values. For example, software can simply check the elapsed number of minutes by reading one register. Alternately, it can read the complete time of day, including subseconds, in only four registers. The calendar stores its data in binary form. While this requires software translation, it allows complete flexibility as to the exact value. A user can start the calendar with a variety of selections since it is simply a 16-bit binary number of days. This number allows a total range of 179 years beginning from 0000.

selected value, it sets a flag. This will cause an interrupt if enabled, even in Stop mode. The alarm consists of a comparator that matches the user value against the RTC actual value. A user can select a match for one or more of the sub-seconds, seconds, minutes, or hours. This allows an interrupt automatically to occur once per second, once per minute, once per hour, or once per day. Enabling interrupts with no match will generate an interrupt 256 times per second.

Software enables the timekeeper oscillator using the RTC Enable bit in the RTC Control register (F9h). This starts the clock. It can disable the oscillator to preserve the life of the backup energy-source if unneeded. Values in the RTC Control register are maintained by the backup source through power failure. Once enabled, the RTC maintains time for the life of the backup source even when  $V_{CC}$  is removed.

The RTC will maintain an accuracy of  $\pm 2$  minutes per month at 25°C. Under no circumstances are negative voltages, of any amplitude, allowed on any pin while the device is in data retention mode ( $V_{CC} < V_{BAT}$ ). Negative voltages will shorten battery life, possibly corrupting the contents of internal SRAM and the RTC.

REAL TIME CLOCK Figure 2



## NONVOLATILE RAM

The 1K x 8 on-chip SRAM can be nonvolatile. An external backup energy-source will maintain the SRAM contents through power failure. This allows the DS87C530 to log data or to store configuration settings. Internal switching circuits will detect the loss of  $V_{CC}$  and switch SRAM power to the backup source on the  $V_{BAT}$  pin. The 256 bytes of direct RAM are not affected by this circuit and are volatile.

## CRYSTAL AND BACKUP SOURCES

To use the unique functions of the DS87C530, two external components are needed. These are a 32.768 KHz timekeeping crystal and a backup energy-source. The following describes guidelines for choosing these devices.

### Timekeeping Crystal

The DS87C530 can use a standard 32.768 KHz crystal as the RTC time base. There are two versions of standard crystals available, with 6 pF and 12.5 pF load capacitance. The tradeoff is that the 6 pF uses less power, giving longer life while  $V_{CC}$  is off, but is more sensitive to noise and board layout. The 12.5 pF crystal uses more power, giving a shorter battery backed life, but produces a more robust oscillator. Bit 6 in the RTC

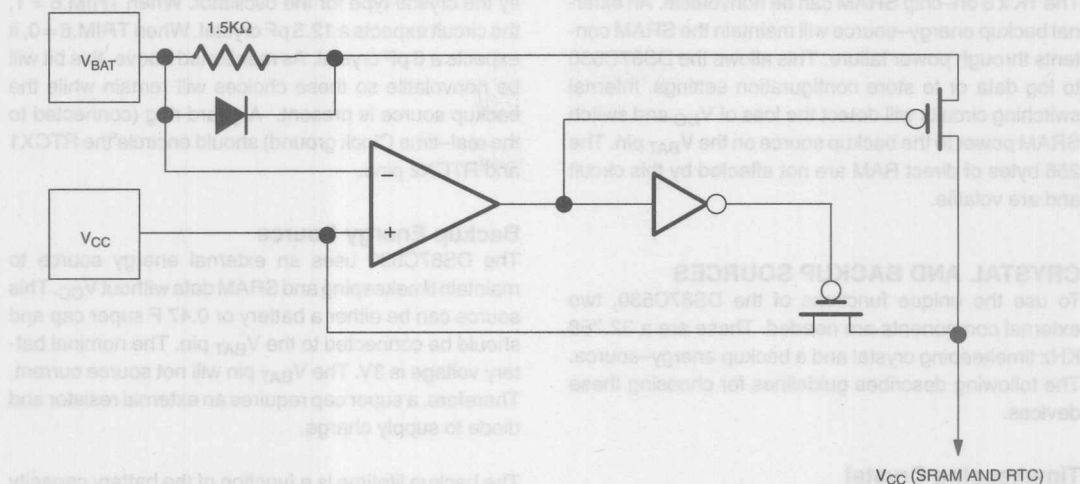
Trim register (TRIM; 96h) must be programmed to specify the crystal type for the oscillator. When TRIM.6 = 1, the circuit expects a 12.5 pF crystal. When TRIM.6 = 0, it expects a 6 pF crystal. As mentioned above, this bit will be nonvolatile so these choices will remain while the backup source is present. A guard ring (connected to the real-time Clock ground) should encircle the RTCX1 and RTCX2 pins.

### Backup Energy Source

The DS87C530 uses an external energy source to maintain timekeeping and SRAM data without  $V_{CC}$ . This source can be either a battery or 0.47 F super cap and should be connected to the  $V_{BAT}$  pin. The nominal battery voltage is 3V. The  $V_{BAT}$  pin will not source current. Therefore, a super cap requires an external resistor and diode to supply charge.

The backup lifetime is a function of the battery capacity and the data retention current drain. This drain is specified in the electrical specifications. The circuit loads the  $V_{BAT}$  only when  $V_{CC}$  has fallen below  $V_{BAT}$ . Thus the actual lifetime depends not only on the current and battery capacity, but also on the portion of time without power. A very small lithium cell provides a lifetime of more than 10 years.

### INTERNAL BACKUP CIRCUIT Figure 3



### IMPORTANT APPLICATION NOTE

The pins on the DS87C530 are generally as resilient as other CMOS circuits. They have no unusual susceptibility to electrostatic discharge (ESD) or other electrical transients. **However, no pin on the DS87C530 should ever be taken to a voltage below ground.** Negative voltages on any pin can turn on internal parasitic diodes that draw current directly from the battery. If a device pin is connected to the "outside world" where it may be handled or come in contact with electrical noise, protection should be added to prevent the device pin from going below -0.3V. Some power supplies can give a small undershoot on power up, which should be prevented. Application Note 93, "Design Guidelines for Microcontrollers Incorporating NVRAM", discusses how to protect the DS87C530 against these conditions.

## MEMORY RESOURCES

Like the 8051, the DS87C530 uses three memory areas. These are program (ROM), data (RAM), and scratchpad RAM (registers). The DS87C530 contains on-chip quantities of all three areas.

The total memory configuration of the DS87C530 is 16KB of ROM, 1KB of data SRAM and 256 bytes of scratchpad or direct RAM. The 1KB of data space SRAM is read/write accessible and is memory mapped. This on-chip SRAM is reached by the MOVX instruction. It is not used for executable memory. The scratchpad area is 256 bytes of register mapped RAM and is identical to the RAM found on the 80C52. There is no conflict or overlap among the 256 bytes and the 1KB as they use different addressing modes and separate instructions.

## PROGRAM MEMORY ACCESS

On-chip ROM begins at address 0000h and is contiguous through 3FFFh (16KB). Exceeding the maximum address of on-chip ROM will cause the DS87C530 to access off-chip memory. However, the maximum on-chip decoded address is selectable by software using the ROMSIZE™ feature. Software can cause the DS87C530 to behave like a device with less on-chip memory. This is beneficial when overlapping external memory, such as Flash, is used.

The maximum memory size is dynamically variable. Thus a portion of memory can be removed from the memory map to access off-chip memory, then restored to access on-chip memory. In fact, all of the on-chip memory can be removed from the memory map allowing the full 64KB memory space to be addressed from off-chip memory. ROM addresses that are larger than the selected maximum are automatically fetched from outside the part via Ports 0 and 2. A depiction of the ROM memory map is shown in Figure 4.

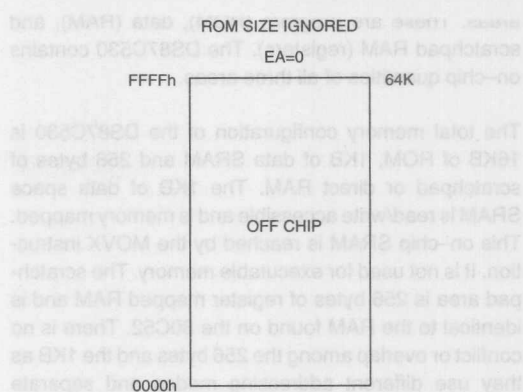
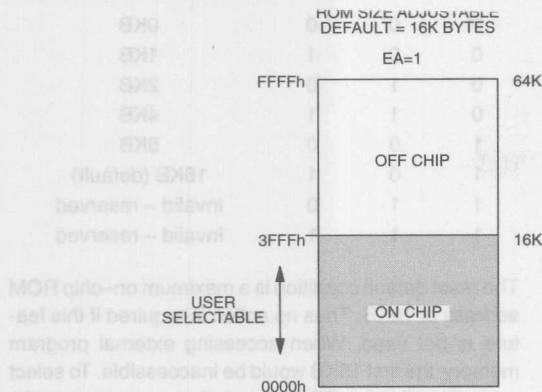
The ROMSIZE register is used to select the maximum on-chip decoded address for ROM. Bits RMS2, RMS1, RMS0 have the following affect.

RMS2	RMS1	RMS0	Maximum on-chip ROM Address
0	0	0	0KB
0	0	1	1KB
0	1	0	2KB
0	1	1	4KB
1	0	0	8KB
1	0	1	16KB (default)
1	1	0	Invalid – reserved
1	1	1	Invalid – reserved

The reset default condition is a maximum on-chip ROM address of 16KB. Thus no action is required if this feature is not used. When accessing external program memory, the first 16KB would be inaccessible. To select a smaller effective ROM size, software must alter bits RMS2–RMS0. Altering these bits requires a Timed Access procedure as explained below.

Care should be taken so that changing the ROMSIZE register does not corrupt program execution. For example, assume that a DS87C520 is executing instructions from internal program memory near the 12KB boundary (~3000h) and that the ROMSIZE register is currently configured for a 16KB internal program space. If software reconfigures the ROMSIZE register to 4KB (0000h–0FFFh) in the current state, the device will immediately jump to external program execution because program code from 4KB to 16KB (1000h–3FFFh) is no longer located on-chip. This could result in code misalignment and execution of an invalid instruction. The recommended method is to modify the ROMSIZE register from a location in memory that will be internal (or external) both before and after the operation. In the above example, the instruction which modifies the ROMSIZE register should be located below the 4KB (1000h) boundary, so that it will be unaffected by the memory modification. The same precaution should be applied if the internal program memory size is modified while executing from external program memory.

Off-chip memory is accessed using the multiplexed address/data bus on P0 and the MSB address on P2. While serving as a memory bus, these pins are not I/O ports. This convention follows the standard 8051 method of expanding on-chip memory. Off-chip ROM access also occurs if the EA pin is a logic 0. EA overrides all bit settings. The PSEN signal will go active (low) to serve as a chip enable or output enable when Ports 0 and 2 fetch from external ROM.



## DATA MEMORY ACCESS

Unlike many 8051 derivatives, the DS87C530 contains on-chip data memory. It also contains the standard 256 bytes of RAM accessed by direct instructions. These areas are separate. The MOVX instruction accesses the on-chip data memory. Although physically on-chip, software treats this area as though it was located off-chip. The 1KB of SRAM is between address 0000h and 03FFh.

Access to the on-chip data RAM is optional under software control. When enabled by software, the data SRAM is between 0000h and 03FFh. Any MOVX instruction that uses this area will go to the on-chip RAM while enabled. MOVX addresses greater than 03FFh automatically go to external memory through Ports 0 and 2.

When disabled, the 1KB memory area is transparent to the system memory map. Any MOVX directed to the space between 0000h and FFFFh goes to the expanded bus on Ports 0 and 2. This also is the default condition. This default allows the DS87C530 to drop into an existing system that uses these addresses for other hardware and still have full compatibility.

The on-chip data area is software selectable using two bits in the Power Management Register at location C4h. This selection is dynamically programmable. Thus access to the on-chip area becomes transparent to reach off-chip devices at the same addresses. The control bits are DME1 (PMR.1) and DME0 (PMR.0). They have the following operation:

**DATA MEMORY ACCESS CONTROL Table 3**

DME1	DME0	DATA MEMORY ADDRESS	MEMORY FUNCTION
0	0	0000h – FFFFh	External Data Memory *Default condition
0	1	0000h – 03FFh 0400h – FFFFh	Internal SRAM Data Memory External Data Memory
1	0	Reserved	Reserved
1	1	0000h – 03FFh 0400h – FFFBh FFFC FFFDh–FFFFh	Internal SRAM Data Memory Reserved – no external access Read access to the status of lock bits Reserved – no external access

Notes on the status byte read at FFFCh with DME1, 0 = 1, 1: Bits 2–0 reflect the programmed status of the security lock bits LB2–LB0. They are individually set to a logic 1 to correspond to a security lock bit that has been programmed. These status bits allow software to verify that the part has been locked before running if desired. The bits are read only.



## STRETCH MEMORY CYCLE

The DS87C530 allows software to adjust the speed of off-chip data memory access. The micro is capable of performing the MOVX in as few as two instruction cycles. The on-chip SRAM uses this speed and any MOVX instruction directed internally uses two cycles. However, the time can be stretched for interface to external devices. This allows access to both fast memory and slow memory or peripherals with no glue logic. Even in high-speed systems, it may not be necessary or desirable to perform off-chip data memory access at full speed. In addition, there are a variety of memory mapped peripherals such as LCDs or UARTs that are slow.

The Stretch MOVX is controlled by the Clock Control Register at SFR location 8Eh as described below. It allows the user to select a Stretch value between zero and seven. A Stretch of zero will result in a two machine cycle MOVX. A Stretch of seven will result in a MOVX of nine machine cycles. Software can dynamically change this value depending on the particular memory or peripheral.

On reset, the Stretch value will default to a one resulting in a three cycle MOVX for any external access. There-

fore, off-chip RAM access is not at full speed. This is a convenience to existing designs that may not have fast RAM in place. Internal SRAM access is always at full speed regardless of the Stretch setting. When desiring maximum speed, software should select a Stretch value of zero. When using very slow RAM or peripherals, select a larger Stretch value. Note that this affects data memory only and the only way to slow program memory (ROM) access is to use a slower crystal.

Using a Stretch value between one and seven causes the microcontroller to stretch the read/write strobe and all related timing. Also, setup and hold times are increased by 1 clock when using any Stretch greater than 0. This results in a wider read/write strobe and relaxed interface timing, allowing more time for memory/peripherals to respond. The timing of the variable speed MOVX is in the Electrical Specifications. Table 4 shows the resulting strobe widths for each Stretch value. The memory Stretch uses the Clock Control Special Function Register at SFR location 8Eh. The Stretch value is selected using bits CKCON.2-0. In the table, these bits are referred to as M2 through M0. The first Stretch (default) allows the use of common 120 ns RAMs without dramatically lengthening the memory access.

**DATA MEMORY CYCLE STRETCH VALUES** Table 4

CKCON.2-0			MEMORY CYCLES	RD OR WR STROBE WIDTH IN CLOCKS	STROBE WIDTH TIME @ 33 MHz
M2	M1	M0			
0	0	0	2 (forced internal)	2	60 ns
0	0	1	3 (default external)	4	121 ns
0	1	0	4	8	242 ns
0	1	1	5	12	364 ns
1	0	0	6	16	485 ns
1	0	1	7	20	606 ns
1	1	0	8	24	727 ns
1	1	1	9	28	848 ns

## DUAL DATA POINTER

The timing of block moves of data memory is faster using the DS87C530 Dual Data Pointer (DPTR). The standard 8051 DPTR is a 16-bit value that is used to address off-chip data RAM or peripherals. In the DS87C530, the standard data pointer is called DPTR, located at SFR addresses 82h and 83h. These are the standard locations. Using DPTR requires no modification of standard code. The new DPTR at SFR 84h and 85h is called DPTR1. The DPTR Select bit (DPS) chooses the active pointer. Its location is the lsb of the SFR location 86h. No other bits in register 86h have any

effect and are 0. The user switches between data pointers by toggling the lsb of register 86h. The increment (INC) instruction is the fastest way to accomplish this. All DPTR-related instructions use the currently selected DPTR for any activity. Therefore it takes only one instruction to switch from a source to a destination address. Using the Dual Data Pointer saves code from needing to save source and destination addresses when doing a block move. The software simply switches between DPTR and 1 once software loads them. The relevant register locations are as follows.

DPL	82h	Low byte original DPTR
DPH	83h	High byte original DPTR
DPL1	84h	Low byte new DPTR
DPH1	85h	High byte new DPTR
DPS	86h	DPTR Select (lsb)

**POWER MANAGEMENT**

Along with the standard Idle and power down (Stop) modes of the standard 80C52, the DS87C530 provides a new Power Management Mode. This mode allows the processor to continue functioning, yet to save power compared with full operation. The DS87C530 also features several enhancements to Stop mode that make it more useful.

**POWER MANAGEMENT MODE (PMM)**

Power Management Mode offers a complete scheme of reduced internal clock speeds that allow the CPU to run software but to use substantially less power. During default operation, the DS87C530 uses four clocks per machine cycle. Thus the instruction cycle rate is (Clock/4). At 33 MHz crystal speed, the instruction cycle speed is 8.25 MHz (33/4). In PMM, the microcontroller

continues to operate but uses an internally divided version of the clock source. This creates a lower power state without external components. It offers a choice of two reduced instruction cycle speeds (and two clock sources – discussed below). The speeds are (Clock/64) and (Clock/1024).

Software is the only mechanism to invoke the PMM. Table 5 illustrates the instruction cycle rate in PMM for several common crystal frequencies. Since power consumption is a direct function of operating speed, PMM 1 eliminates most of the power consumption while still allowing a reasonable speed of processing. PMM 2 runs very slowly and provides the lowest power consumption without stopping the CPU. This is illustrated in Table 6.

Note that PMM provides a lower power condition than Idle mode. This is because in Idle, all clocked functions such as timers run at a rate of crystal divided by 4. Since wake-up from PMM is as fast as or faster than from Idle and PMM allows the CPU to operate (even if doing NOPs), there is little reason to use Idle mode in new designs.

**INSTRUCTION CYCLE RATE** Table 5

CRYSTAL SPEED	FULL OPERATION (4 CLOCKS)	PMM 1 (64 CLOCKS)	PMM 2 (1024 CLOCKS)
1.8432 MHz	460.8 KHz	28.8 KHz	1.8 KHz
11.0592 MHz	2.765 MHz	172.8 KHz	10.8 KHz
22 MHz	5.53 MHz	345.6 KHz	21.6 KHz
25 MHz	6.25 MHz	390.6 KHz	24.4 KHz
33 MHz	8.25 MHz	515.6 KHz	32.2 KHz

**OPERATING CURRENT ESTIMATES IN PMM** Table 6

CRYSTAL SPEED	FULL OPERATION (4 CLOCKS)	PMM 1 (64 CLOCKS)	PMM 2 (1024 CLOCKS)
1.8432 MHz	3.1 mA	1.2 mA	1.0 mA
3.57 MHz	5.3 mA	1.6 mA	1.1 mA
11.0592 MHz	15.5 mA	4.8 mA	4.0 mA
16 MHz	21 mA	7.1 mA	6.0 mA
22 MHz	25.5 mA	8.3 mA	6.5 mA
25 MHz	31 mA	9.7 mA	8.0 mA
33 MHz	36 mA	12.0 mA	10.0 mA

## CRYSTALESS PMM

A major component of power consumption in PMM is the crystal amplifier circuit. The DS87C530 allows the user to switch CPU operation to an internal ring oscillator and turn off the crystal amplifier. The CPU would then have a clock source of approximately 2–4 MHz, divided by either 4, 64, or 1024. The ring is not accurate, so software can not perform precision timing. However, this mode allows an additional saving of between 0.5 and 6.0 mA depending on the actual crystal frequency. While this saving is of little use when running at 4 clocks per instruction cycle, it makes a major contribution when running in PMM1 or PMM2.

## PMM OPERATION

Software invokes the PMM by setting the appropriate bits in the SFR area. The basic choices are divider speed and clock source. There are three speeds (4, 64, and 1024) and two clock sources (crystal, ring). Both the decisions and the controls are separate. Software will typically select the clock speed first. Then, it will perform the switch to ring operation if desired. Lastly, software can disable the crystal amplifier if desired.

There are two ways of exiting PMM. Software can remove the condition by reversing the procedure that invoked PMM or hardware can (optionally) remove it. To resume operation at a divide by 4 rate under software control, simply select 4 clocks per cycle, then crystal based operation if relevant. When disabling the crystal as the time base in favor of the ring oscillator, there are timing restrictions associated with restarting the crystal operation. Details are described below.

There are three registers containing bits that are concerned with PMM functions. They are Power Management Register (PMR; C4h), Status (STATUS; C5h), and External Interrupt Flag (EXIF; 91h).

## Clock Divider

Software can select the instruction cycle rate by selecting bits CD1 (PMR.7) and CD0 (PMR.6) as follows:

CD1	CD0	Cycle rate
0	0	Reserved
0	1	4 clocks (default)
1	0	64 clocks
1	1	1024 clocks

The selection of instruction cycle rate will take effect after a delay of one instruction cycle. Note that the clock divider choice applies to all functions including timers. Since baud rates are altered, it will be difficult to conduct serial communication while in PMM. There are minor restrictions on accessing the clock selection bits. The processor must be running in a 4 clock state to select either 64 (PMM1) or 1024 (PMM2) clocks. This means software cannot go directly from PMM1 to PMM2 or visa versa. It must return to a 4 clock rate first.

## Switchback

To return to a 4 clock rate from PMM, software can simply select the CD1 and CD0 clock control bits to the 4 clocks per cycle state. However, the DS87C530 provides several hardware alternatives for automatic Switchback. If Switchback is enabled, then the DS87C530 will automatically return to a 4 clock per cycle speed when an interrupt occurs from an enabled, valid external interrupt source. A Switchback will also occur when a UART detects the beginning of a serial start bit if the serial receiver is enabled (REN=1). Note the beginning of a start bit does not generate an interrupt; this occurs on reception of a complete serial word. The automatic Switchback on detection of a start bit allows hardware to correct baud rates in time for a proper serial reception. A switchback will also occur when a byte is written to the SBUF0 or SBUF1 for transmission.

Switchback is enabled by setting the SWB bit (PMR.5) to a 1 in software. For an external interrupt, Switchback will occur only if the interrupt source could really generate the interrupt. For example, if INT0 is enabled but has a low priority setting, then Switchback will not occur on INT0 if the CPU is servicing a high priority interrupt.

## Status

Information in the Status register assists decisions about switching into PMM. This register contains information about the level of active interrupts and the activity on the serial ports.

The DS87C530 supports three levels of interrupt priority. These levels are Power-fail, High, and Low. Bits STATUS.7–5 indicate the service status of each level. If PIP (Power-fail Interrupt Priority; STATUS.7) is a 1, then the processor is servicing this level. If either HIP

(High Interrupt Priority; STATUS.6) or LIP (Low Interrupt Priority; STATUS.5) is high, then the corresponding level is in service.

Software should not rely on a lower priority level interrupt source to remove PMM (Switchback) when a higher level is in service. Check the current priority service level before entering PMM. If the current service level locks out a desired Switchback source, then it would be advisable to wait until this condition clears before entering PMM.

Alternately, software can prevent an undesired exit from PMM by entering a low priority interrupt service level before entering PMM. This will prevent other low priority interrupts from causing a Switchback.

Status also contains information about the state of the serial ports. Serial Port Zero Receive Activity (SPRA0; STATUS.0) indicates a serial word is being received on Serial Port 0 when this bit is set to a 1. Serial Port Zero Transmit Activity (SPTA0; STATUS.1) indicates that the serial port is still shifting out a serial transmission. STATUS.2 and STATUS.3 provide the same information for Serial Port 1, respectively. These bits should be interrogated before entering PMM1 or PMM2 to ensure that no serial port operations are in progress. Changing the clock divisor rate during a serial transmission or reception will corrupt the operation.

### Crystal/Ring Operation

The DS87C530 allows software to choose the clock source as an independent selection from the instruction cycle rate. The user can select crystal-based or ring oscillator-based operation under software control. Power-on reset default is the crystal (or external clock) source. The ring may save power depending on the actual crystal speed. To save still more power, software can then disable the crystal amplifier. This process requires two steps. Reversing the process also requires two steps.

The XT/RG bit (EXIF.3) selects the crystal or ring as the clock source. Setting XT/RG = 1 selects the crystal. Set-

ting XT/RG = 0 selects the ring. The RGMD (EXIF.2) bit serves as a status bit by indicating the active clock source. RGMD = 0 indicates the CPU is running from the crystal. RGMD = 1 indicates it is running from the ring. When operating from the ring, disable the crystal amplifier by setting the XTOFF bit (PMR.3) to a 1. This can only be done when XT/RG = 0.

When changing the clock source, the selection will take effect after a one instruction cycle delay. This applies to changes from crystal to ring and vice versa. However, this assumes that the crystal amplifier is running. In most cases, when the ring is active, software previously disabled the crystal to save power. If ring operation is being used and the system must switch to crystal operation, the crystal must first be enabled. Set the XTOFF bit to a 0. At this time, the crystal oscillation will begin. The DS87C530 then provides a warm-up delay to make certain that the frequency is stable. Hardware will set the XTUP bit (STATUS.4) to a 1 when the crystal is ready for use. Then software should write XT/RG to a 1 to begin operating from the crystal. Hardware prevents writing XT/RG to a 1 before XTUP = 1. The delay between XTOFF = 0 and XTUP = 1 will be 65,536 crystal clocks in addition to the crystal cycle startup time.

Switchback has no effect on the clock source. If software selects a reduced clock divider and enables the ring, a Switchback will only restore the divider speed. The ring will remain as the time base until altered by software. If there is serial activity, Switchback usually occurs with enough time to create proper baud rates. This is not true if the crystal is off and the CPU is running from the ring. If sending a serial character that wakes the system from crystaless PMM, then it should be a dummy character of no importance with a subsequent delay for crystal startup.

The following table is a summary of the bits relating to PMM and its operation. The flow chart below illustrates a typical decision set associated with PMM.

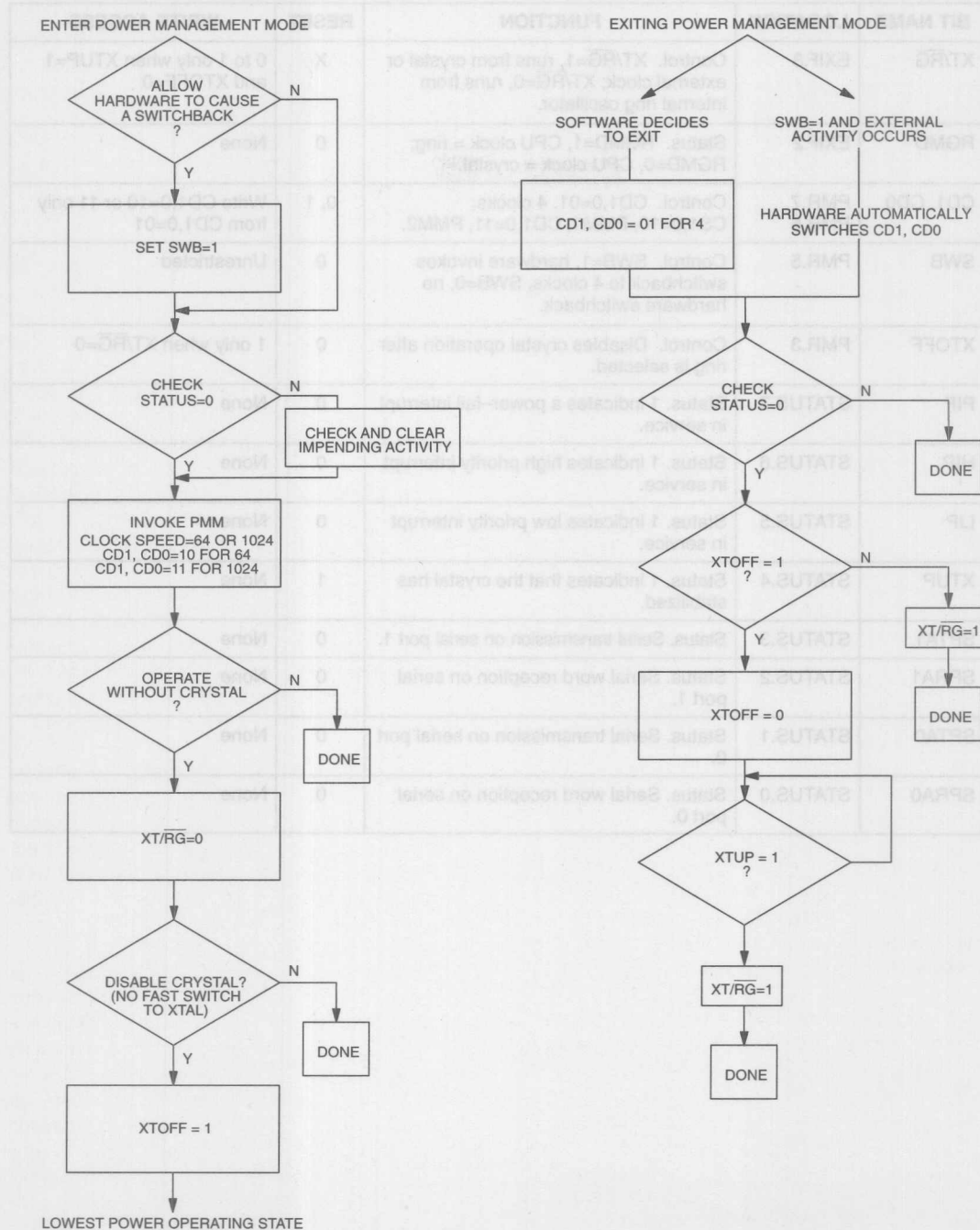
CDT	CD0	Cycle rate
0	0	Reserved
0	1	4 clocks (main)
1	0	64 clocks
1	1	1024 clocks



**PMM CONTROL AND STATUS BIT SUMMARY** Table 7

BIT NAME	LOCATION	FUNCTION	RESET	WRITE ACCESS
XT/RG	EXIF.3	Control. XT/RG=1, runs from crystal or external clock; XT/RG=0, runs from internal ring oscillator.	X	0 to 1 only when XTUP=1 and XTOFF=0
RGMD	EXIF.2	Status. RGMD=1, CPU clock = ring; RGMD=0, CPU clock = crystal.	0	None
CD1, CD0	PMR.7, PMR.6	Control. CD1,0=01, 4 clocks; CS1,0=10, PMM1; CD1,0=11, PMM2.	0, 1	Write CD1,0=10 or 11 only from CD1,0=01
SWB	PMR.5	Control. SWB=1, hardware invokes switchback to 4 clocks, SWB=0, no hardware switchback.	0	Unrestricted
XTOFF	PMR.3	Control. Disables crystal operation after ring is selected.	0	1 only when XT/RG=0
PIP	STATUS.7	Status. 1 indicates a power-fail interrupt in service.	0	None
HIP	STATUS.6	Status. 1 indicates high priority interrupt in service.	0	None
LIP	STATUS.5	Status. 1 indicates low priority interrupt in service.	0	None
XTUP	STATUS.4	Status. 1 indicates that the crystal has stabilized.	1	None
SPTA1	STATUS.3	Status. Serial transmission on serial port 1.	0	None
SPRA1	STATUS.2	Status. Serial word reception on serial port 1.	0	None
SPTA0	STATUS.1	Status. Serial transmission on serial port 0.	0	None
SPRA0	STATUS.0	Status. Serial word reception on serial port 0.	0	None



**INVOKING AND CLEARING PMM** Figure 3

## IDLE MODE

Setting the lsb of the Power Control register (PCON; 87h) invokes the Idle mode. Idle will leave internal clocks, serial ports and timers running. Power consumption drops because the CPU is not active. Since clocks are running, the Idle power consumption is a function of crystal frequency. It should be approximately 1/2 of the operational power at a given frequency. The CPU can exit the Idle state with any interrupt or a reset. Idle is available for backward software compatibility. The system can now reduce power consumption to below Idle levels by using PMM1 or PMM2 and running NOPs.

## STOP MODE ENHANCEMENTS

Setting bit 1 of the Power Control register (PCON; 87h) invokes the Stop mode. Stop mode is the lowest power state since it turns off all internal clocking. The  $I_{CC}$  of a standard Stop mode is approximately 1  $\mu$ A but is specified in the Electrical Specifications. The CPU will exit Stop mode from an external interrupt or a reset condition. Internally generated interrupts (timer, serial port, watchdog) are not useful since they require clocking activity. One exception is that a real-time clock interrupt can cause the device to exit Stop mode. This provides a very power efficient way of performing infrequent yet periodic tasks.

The DS87C530 provides two enhancements to the Stop mode. As documented below, the DS87C530 provides a band-gap reference to determine Power-fail Interrupt and Reset thresholds. The default state is that the band-gap reference is off while in Stop mode. This allows the extremely low power state mentioned above. A user can optionally choose to have the band-gap enabled during Stop mode. With the band-gap reference enabled, PFI and Power-fail reset are functional and are a valid means for leaving Stop mode. This allows software to detect and compensate for a brown-out or power supply sag, even when in Stop mode.

In Stop mode with the band-gap enabled,  $I_{CC}$  will be approximately 50  $\mu$ A compared with 1  $\mu$ A with the band-gap off. If a user does not require a Power-fail Reset or Interrupt while in Stop mode, the band-gap can remain disabled. Only the most power sensitive applications should turn off the band-gap, as this results in an uncontrolled power down condition.

The control of the band-gap reference is located in the Extended Interrupt Flag register (EXIF; 91h). Setting BGS (EXIF.0) to a 1 will keep the band-gap reference

enabled during Stop mode. The default or reset condition is with the bit at a logic 0. This results in the band-gap being off during Stop mode. Note that this bit has no control of the reference during full power, PMM, or Idle modes.

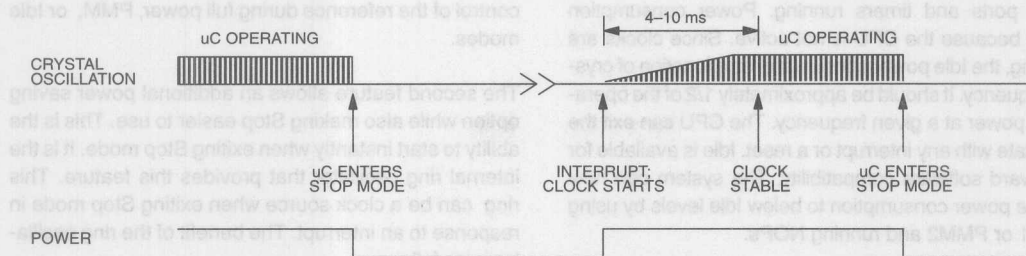
The second feature allows an additional power saving option while also making Stop easier to use. This is the ability to start instantly when exiting Stop mode. It is the internal ring oscillator that provides this feature. This ring can be a clock source when exiting Stop mode in response to an interrupt. The benefit of the ring oscillator is as follows.

Using Stop mode turns off the crystal oscillator and all internal clocks to save power. This requires that the oscillator be restarted when exiting Stop mode. Actual start-up time is crystal dependent, but is normally at least 4 ms. A common recommendation is 10 ms. In an application that will wake-up, perform a short operation, then return to sleep, the crystal start-up can be longer than the real transaction. However, the ring oscillator will start instantly. Running from the ring, the user can perform a simple operation and return to sleep before the crystal has even started. If a user selects the ring to provide the start-up clock and the processor remains running, hardware will automatically switch to the crystal once a power-on reset interval (65536 clocks) has expired. Hardware uses this value to assure proper crystal start even though power is not being cycled.

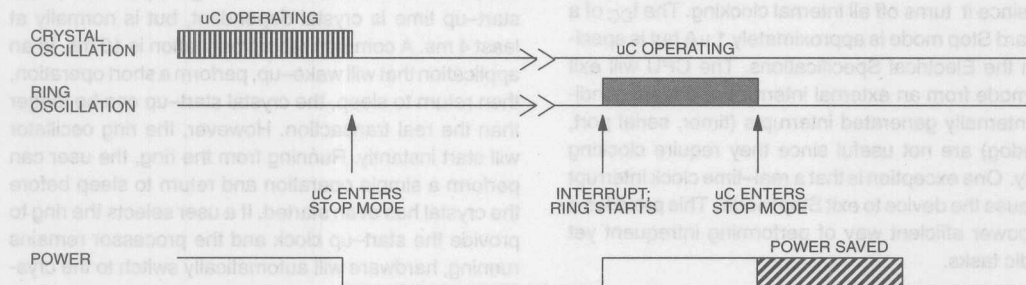
The ring oscillator runs at approximately 2–4 MHz but will not be a precise value. Do not conduct real-time precision operations (including serial communication) during this ring period. Figure 4 shows how the operation would compare when using the ring, and when starting up normally. The default state is to exit Stop mode without using the ring oscillator.

The RGSL – Ring Select bit at EXIF.1 (EXIF; 91h) controls this function. When RGSL = 1, the CPU will use the ring oscillator to exit Stop mode quickly. As mentioned above, the processor will automatically switch from the ring to the crystal after a delay of 65,536 crystal clocks. For a 3.57 MHz crystal, this is approximately 18 ms. The processor sets a flag called RGMD – Ring Mode, located at EXIF.2, that tells software that the ring is being used. The bit will be a logic 1 when the ring is in use. Attempt no serial communication or precision timing while this bit is set, since the operating frequency is not precise.

## STOP MODE WITHOUT RING STARTUP



## STOP MODE WITH RING STARTUP



Note: Diagram assumes that the operation following Stop requires less than 18 ms to complete.

## EMI REDUCTION

The DS87C530 allows software to reduce EMI. One of the major contributors to radiated noise in an 8051 based system is the toggling of ALE. The DS87C530 allows software to disable ALE when not used by setting the ALEOFF (PMR.2) bit to a 1. When ALEOFF = 1, ALE will still toggle during an off-chip MOVX. However, ALE will remain in a static when performing on-chip memory access. The default state of ALEOFF = 0 so ALE toggles with every instruction cycle.

## PERIPHERAL OVERVIEW

The DS87C530 provides several of the most commonly needed peripheral functions in microcomputer-based systems. These new functions include a second serial port, Power-fail Reset, Power-fail Interrupt, and a programmable Watchdog Timer. These are described

below, and more details are available in the High-Speed Microcontroller User's Guide.

## SERIAL PORTS

The DS87C530 provides a serial port (UART) that is identical to the 80C52. In addition it includes a second hardware serial port that is a full duplicate of the standard one. This port optionally uses pins P1.2 (RXD1) and P1.3 (TXD1). It has duplicate control functions included in new SFR locations.

Both ports can operate simultaneously but can be at different baud rates or even in different modes. The second serial port has similar control registers (SCON1; C0h, SBUF1; C1h) to the original. The new serial port can only use Timer 1 for timer generated baud rates.

## TIMER RATE CONTROL

There is one important difference between the DS87C530 and 8051 regarding timers. The original 8051 used 12 clocks per cycle for timers as well as for machine cycles. The DS87C530 architecture normally uses 4 clocks per machine cycle. However, in the area of timers and serial ports, the DS87C530 will default to 12 clocks per cycle on reset. This allows existing code with real-time dependencies such as baud rates to operate properly.

If an application needs higher speed timers or serial baud rates, the user can select individual timers to run at the 4 clock rate. The Clock Control register (CKCON; 8Eh) determines these timer speeds. When the relevant CKCON bit is a logic 1, the DS87C530 uses 4 clocks per cycle to generate timer speeds. When the bit is a 0, the DS87C530 uses 12 clocks for timer speeds. The reset condition is a 0. CKCON.5 selects the speed of Timer 2. CKCON.4 selects Timer 1 and CKCON.3 selects Timer 0. Unless a user desires very fast timing, it is unnecessary to alter these bits. Note that the timer controls are independent.

## POWER FAIL RESET

The DS87C530 uses a precision band-gap voltage reference to decide if  $V_{CC}$  is out of tolerance. While powering up, the internal monitor circuit maintains a reset state until  $V_{CC}$  rises above the  $V_{RST}$  level. Once above this level, the monitor enables the crystal oscillator and counts 65536 clocks. It then exits the reset state. This power-on reset (POR) interval allows time for the oscillator to stabilize.

A system needs no external components to generate a power-related reset. Anytime  $V_{CC}$  drops below  $V_{RST}$ , as in power-failure or a power drop, the monitor will generate and hold a reset. It occurs automatically, needing no action from the software. Refer to the Electrical Specifications for the exact value of  $V_{RST}$ .

## POWER FAIL INTERRUPT

The voltage reference that sets a precise reset threshold also generates an optional early warning Power-fail Interrupt (PFI). When enabled by software, the processor will vector to program memory address 0033h if  $V_{CC}$  drops below  $V_{PFW}$ . PFI has the highest priority. The PFI enable is in the Watchdog Control SFR (WDCON –

D8h). Setting WDCON.5 to a logic 1 will enable the PFI. Application software can also read the PFI flag at WDCON.4. A PFI condition sets this bit to a 1. The flag is independent of the interrupt enable and software must manually clear it. If the PFI is enabled and the band-gap select bit (BGS) is set, a PFI will bring the device out of Stop mode.

## WATCHDOG TIMER

To prevent software from losing control, the DS87C530 includes a programmable Watchdog Timer. The Watchdog is a free running timer that sets a flag if allowed to reach a preselected time-out. It can be (re)started by software.

A typical application is to select the flag as a reset source. When the Watchdog times out, it sets its flag which generates reset. Software must restart the timer before it reaches its time-out or the processor is reset.

Software can select one of four time-out values. Then, it restarts the timer and enables the reset function. After enabling the reset function, software must then restart the timer before its expiration or hardware will reset the CPU. Both the Watchdog Reset Enable and the Watchdog Restart control bits are protected by a "Timed Access" circuit. This prevents errant software from accidentally clearing the Watchdog. Time-out values are precise since they are a function of the crystal frequency as shown below in Table 8. For reference, the time periods at 33 MHz also are shown.

The Watchdog also provides a useful option for systems that do not require a reset circuit. It will set an interrupt flag 512 clocks before setting the reset flag. Software can optionally enable this interrupt source. The interrupt is independent of the reset. A common use of the interrupt is during debug, to show developers where the Watchdog times out. This indicates where the Watchdog must be restarted by software. The interrupt also can serve as a convenient time-base generator or can wake-up the processor from power saving modes.

The Watchdog function is controlled by the Clock Control (CKCON – 8Eh), Watchdog Control (WDCON – D8h), and Extended Interrupt Enable (EIE – E8h) SFRs. CKCON.7 and CKCON.6 are WD1 and WD0 respectively and they select the Watchdog time-out period as shown in Table 8.



**WATCHDOG TIME-OUT VALUES** Table 8

WD1	WD0	INTERRUPT TIME-OUT	TIME (33 MHz)	RESET TIME-OUT	TIME (33 MHz)
0	0	$2^{17}$ clocks	3.9718 ms	$2^{17} + 512$ clocks	3.9874 ms
0	1	$2^{20}$ clocks	31.77 ms	$2^{20} + 512$ clocks	31.79 ms
1	0	$2^{23}$ clocks	254.20 ms	$2^{23} + 512$ clocks	254.21 ms
1	1	$2^{26}$ clocks	2033.60 ms	$2^{26} + 512$ clocks	2033.62 ms

As shown above, the Watchdog Timer uses the crystal frequency as a time base. A user selects one of four counter values to determine the time-out. These clock counter lengths are  $2^{17} = 131,072$  clocks;  $2^{20} = 1,048,576$ ;  $2^{23} = 8,388,608$  clocks; and  $2^{26} = 67,108,864$  clocks. The times shown in Table 8 above are with a 33 MHz crystal frequency. Once the counter chain has completed a full interrupt count, hardware will set an interrupt flag. Regardless of whether the user enables this interrupt, there are then 512 clocks left until the reset flag is set. Software can enable the interrupt and reset individually. Note that the Watchdog is a free running timer and does not require an enable.

There are five control bits in special function registers that affect the Watchdog Timer and two status flags that report to the user. WDIF (WDCON.3) is the interrupt flag that is set at timer termination when there are 512 clocks remaining until the reset flag is set. WTRF (WDCON.2) is the flag that is set when the timer has completely timed out. This flag is normally associated with a CPU reset and allows software to determine the reset source.

EWT (WDCON.1) is the enable for the Watchdog timer reset function. RWT (WDCON.0) is the bit that software uses to restart the Watchdog Timer. Setting this bit restarts the timer for another full interval. Application software must set this bit before the time-out. Both of these bits are protected by Timed Access discussed below. As mentioned previously, WD1 and 0 (CKCON.7 and 6) select the time-out. Finally, the user can enable the Watchdog Interrupt using EWDI (EIE.4). The Special Function Register map is shown above.

## INTERRUPTS

The DS87C530 provides 14 interrupt sources with three priority levels. The Power-fail Interrupt (PFI) has the highest priority. Software can assign high or low priority to other sources. All interrupts that are new to the 8051 family, except for the PFI, have a lower natural priority than the originals.

**INTERRUPT SOURCES AND PRIORITIES** Table 9

NAME	DESCRIPTION	VECTOR	NATURAL PRIORITY	8051/DALLAS
PFI	Power Fail Interrupt	33h	1	DALLAS
INT0	External Interrupt 0	03h	2	8051
TF0	Timer 0	0Bh	3	8051
INT1	External Interrupt 1	13h	4	8051
TF1	Timer 1	1Bh	5	8051
SCON0	TI0 or RI0 from serial port 0	23h	6	8051
TF2	Timer 2	2Bh	7	8051
SCON1	TI1 or RI1 from serial port 1	3Bh	8	DALLAS
INT2	External Interrupt 2	43h	9	DALLAS
INT3	External Interrupt 3	4Bh	10	DALLAS
INT4	External Interrupt 4	53h	11	DALLAS
INT5	External Interrupt 5	5Bh	12	DALLAS
WDTI	Watchdog Time-Out Interrupt	63h	13	DALLAS
RTCI	Real-Time clock Interrupt	6Bh	14	DALLAS



## TIMED ACCESS PROTECTION

It is useful to protect certain SFR bits from an accidental write operation. The Timed Access procedure stops an errant CPU from accidentally changing these bits. It requires that the following instructions precede a write of a protected bit.

```
MOV    0C7h, #0AAh
MOV    0C7h, #55h
```

Writing an AAh then a 55h to the Timed Access register (location C7h) opens a 3 cycle window for write access. The window allows software to modify a protected bit(s). If these instructions do not immediately precede the write operation, then the write will not take effect. The protected bits are:

EXIF.0	BGS	Band-gap Select
WDCON.6	POR	Power-on Reset flag
WDCON.1	EWT	Enable Watchdog Reset
WDCON.0	RWT	Restart Watchdog
WDCON.3	WDIF	Watchdog Interrupt Flag
ROMSIZE.2	RMS2	ROM size select 2
ROMSIZE.1	RMS1	ROM size select 1
ROMSIZE.0	RMS0	ROM size select 0
TRIM.7-0		All RTC trim functions
RTCC.2	RTCWE	RTC Write Enable
RTCC.0	RTCE	RTC Oscillator Enable

## EPROM PROGRAMMING

The DS87C530 follows standards for a 16K byte EPROM version in the 8051 family. It is available in a UV erasable, ceramic windowed package and in plastic packages for one-time user-programmable versions. The part has unique signature information so programmers can support its specific EPROM options.

## PROGRAMMING PROCEDURE

The DS87C530 should run from a clock speed between 4 and 6 MHz when programmed. The programming fixture should apply address information for each byte to the address lines and the data value to the data lines. The control signals must be manipulated as shown in Table 10. The diagram in Figure 5 shows the expected electrical connection for programming. Note that the programmer must apply addresses in demultiplexed fashion to Ports 1 and 2 with data on Port 0. Waveforms and timing are provided in the Electrical Specifications. Program the DS87C530 as follows:

7. Apply the address value,
8. Apply the data value,
9. Select the programming option from Table 10 using the control signals,
10. Increase the voltage on V<sub>PP</sub> from 5V to 12.75V if writing to the EPROM,
11. Pulse the PROG signal five times for EPROM array and 25 times for encryption table, lock bits, and other EPROM bits,
12. Repeat as many times as necessary.

## SECURITY OPTIONS

The DS87C530 employs a standard three-level lock that restricts viewing of the EPROM contents. A 64-byte Encryption Array allows the authorized user to verify memory by presenting the data in encrypted form.

### Lock Bits

The security lock consists of three lock bits. These bits select a total of 4 levels of security. Higher levels provide increasing security but also limit application flexibility. Table 11 shows the security settings. Note that the programmer cannot directly read the state of the security lock. User software has access to this information as described in the Memory section.

### Encryption Array

The Encryption Array allows an authorized user to verify EPROM without allowing the true memory to be dumped. During a verify, each byte is Exclusive NORed (XNOR) with a byte in the Encryption Array. This results in a true representation of the EPROM while the Encryption is unprogrammed (FFh). Once the Encryption Array is programmed in a non-FFh state, the verify value will be encrypted.

For encryption to be effective, the Encryption Array must be unknown to the party that is trying to verify memory. The entire EPROM also should be a non-FFh state or the Encryption Array can be discovered.

The Encryption Array is programmed as shown in Table 10. Note that the programmer can not read the array. Also note that the verify operation always uses the Encryption Array. The array has no impact while FFh. Simply programming the array to a non-FFh state will cause the encryption to function.

## OTHER EPROM OPTIONS

The DS87C530 has user selectable options that must be set before beginning software execution. These options use EPROM bits rather than SFRs.

Program the EPROM selectable options as shown in Table 10. The Option Register sets or reads these selections. The bits in the Option Control Register have the following function:

Bit 7–4 Reserved, program to a 1.

Bit 3 Watchdog POR default. Set=1; Watchdog reset function is disabled on power-up. Set=0; Watchdog reset function is enabled automatically.

Bit 2–0 Reserved. Program to a 1.

## SIGNATURE

The Signature bytes identify the product and programming revision to EPROM programmers. This information is at programming addresses 30h, 31h, and 60h. This information is as follows::

Address	Value	Meaning
30h	DAh	Manufacturer
31h	30h	Model
60h	01h	Extension

**EPROM PROGRAMMING MODES** Table 10

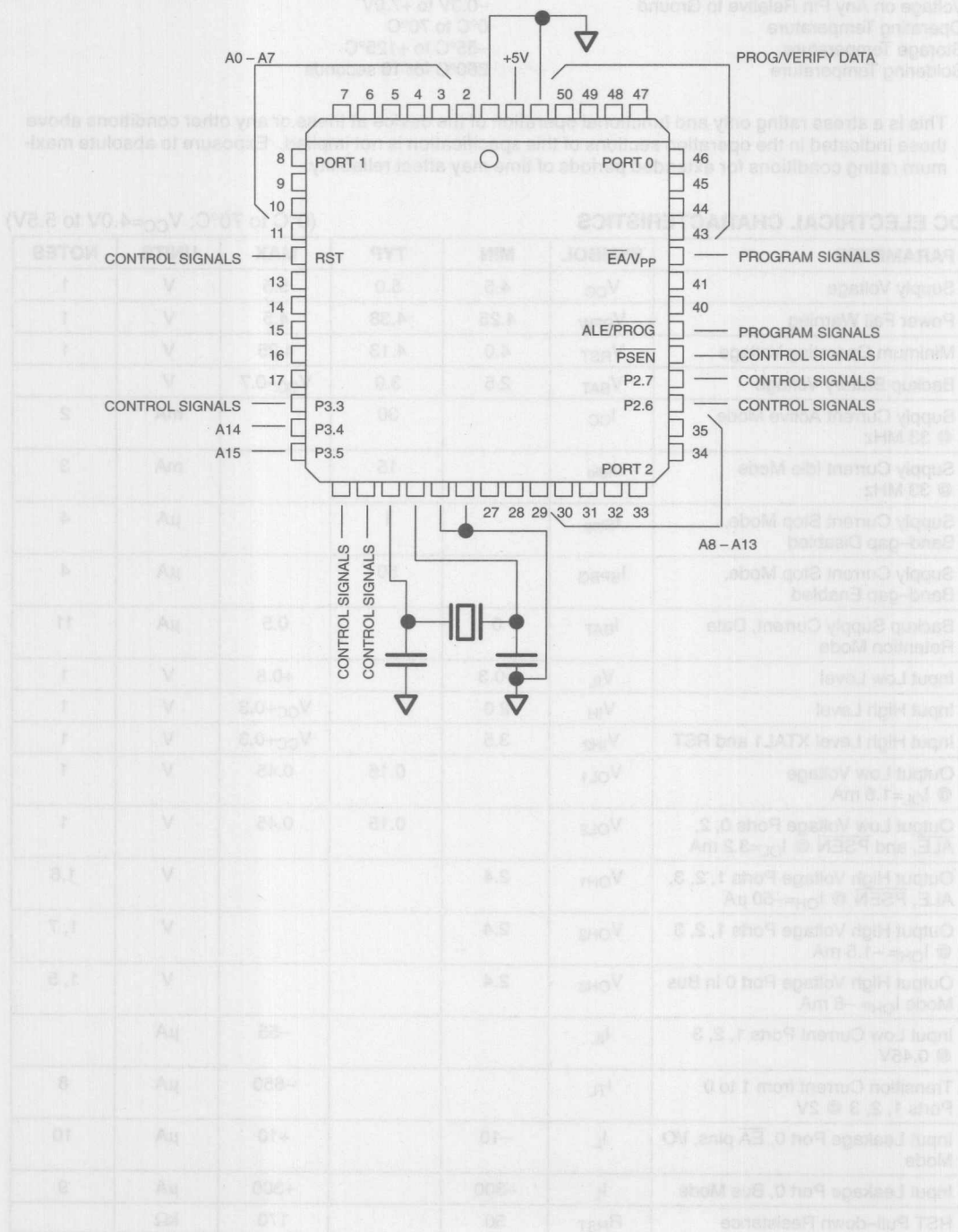
MODE		RST	PSEN	ALE/PROG	EA/VPP	P2.6	P2.7	P3.3	P3.6	P3.7
Program Code Data		H	L	PL	12.75V	L	H	H	H	H
Verify Code Data		H	L	H	H	L	L	L	H	H
Program Encryption Array Address 0–3Fh		H	L	PL	12.75V	L	H	H	L	H
Program Lock Bits	LB1	H	L	PL	12.75V	H	H	H	H	H
	LB2	H	L	PL	12.75V	H	H	H	L	L
	LB3	H	L	PL	12.75V	H	L	H	H	L
Program Option Register Address FCh		H	L	PL	12.75V	L	H	H	L	L
Read Signature or Option Registers 30, 31, 60, FCh		H	L	H	H	L	L	L	L	L

\* PL indicates pulse to a logic low.

**EPROM LOCK BITS** Table 11

LEVEL	LOCK BITS			PROTECTION
	LB1	LB2	LB3	
1	U	U	U	No program lock. Encrypted verify if encryption table was programmed.
2	P	U	U	Prevent MOVX instructions in external memory from reading program bytes in internal memory. EA is sampled and latched on reset. Allow no further programming of EPROM.
3	P	P	U	Level 2 plus no verify operation. Also, prevent MOVX instructions in external memory from reading SRAM (MOVX) in internal memory.
4	P	P	P	Level 3 plus no external execution.

EPROM PROGRAMMING CONFIGURATION Figure 5



**ABSOLUTE MAXIMUM RATINGS\***

Voltage on Any Pin Relative to Ground

-0.3V to +7.0V

Operating Temperature

0°C to 70°C

Storage Temperature

-55°C to +125°C

Soldering Temperature

260°C for 10 seconds

\* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

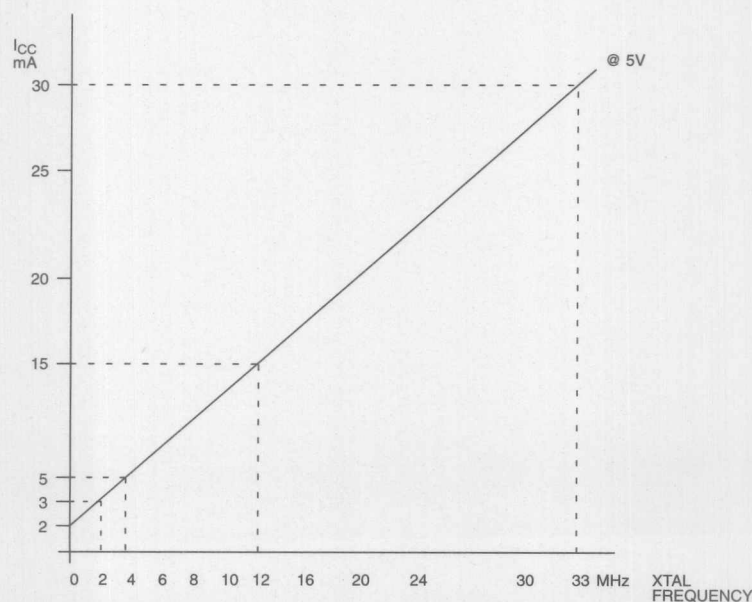
**DC ELECTRICAL CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	$V_{CC}$	4.5	5.0	5.5	V	1
Power Fail Warning	$V_{PFW}$	4.25	4.38	4.5	V	1
Minimum Operating Voltage	$V_{RST}$	4.0	4.13	4.25	V	1
Backup Battery Voltage	$V_{BAT}$	2.5	3.0	$V_{CC}-0.7$	V	
Supply Current Active Mode @ 33 MHz	$I_{CC}$		30		mA	2
Supply Current Idle Mode @ 33 MHz	$I_{Idle}$		15		mA	3
Supply Current Stop Mode, Band-gap Disabled	$I_{Stop}$		1		$\mu A$	4
Supply Current Stop Mode, Band-gap Enabled	$I_{SPBG}$		50		$\mu A$	4
Backup Supply Current, Data Retention Mode	$I_{BAT}$	0		0.5	$\mu A$	11
Input Low Level	$V_{IL}$	-0.3		+0.8	V	1
Input High Level	$V_{IH}$	2.0		$V_{CC}+0.3$	V	1
Input High Level XTAL1 and RST	$V_{IH2}$	3.5		$V_{CC}+0.3$	V	1
Output Low Voltage @ $I_{OL}=1.6$ mA	$V_{OL1}$		0.15	0.45	V	1
Output Low Voltage Ports 0, 2, ALE, and PSEN @ $I_{OL}=3.2$ mA	$V_{OL2}$		0.15	0.45	V	1
Output High Voltage Ports 1, 2, 3, ALE, PSEN @ $I_{OH}=-50$ $\mu A$	$V_{OH1}$	2.4			V	1,6
Output High Voltage Ports 1, 2, 3 @ $I_{OH}=-1.5$ mA	$V_{OH2}$	2.4			V	1,7
Output High Voltage Port 0 in Bus Mode @ $I_{OH}=-8$ mA	$V_{OH3}$	2.4			V	1,5
Input Low Current Ports 1, 2, 3 @ 0.45V	$I_{IL}$			-55	$\mu A$	
Transition Current from 1 to 0 Ports 1, 2, 3 @ 2V	$I_{TL}$			-650	$\mu A$	8
Input Leakage Port 0, E $\bar{A}$ pins, I/O Mode	$I_L$	-10		+10	$\mu A$	10
Input Leakage Port 0, Bus Mode	$I_L$	-300		+300	$\mu A$	9
RST Pull-down Resistance	$R_{RST}$	50		170	k $\Omega$	

**NOTES FOR DC ELECTRICAL CHARACTERISTICS:**

All parameters apply to both commercial and industrial temperature operation unless otherwise noted.

1. All voltages are referenced to ground.
2. Active current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=RST=5.5V$ , all other pins disconnected.
3. Idle mode current is measured with a 33 MHz clock source driving XTAL1,  $V_{CC}=5.5V$ , RST at ground, all other pins disconnected.
4. Stop mode current measured with XTAL1 and RST grounded,  $V_{CC}=5.5V$ , all other pins disconnected. This value is not guaranteed. Users that are sensitive to this specification should contact Dallas Semiconductor for more information.
5. When addressing external memory.
6.  $RST=5.5V$ . This condition mimics operation of pins in I/O mode. Port 0 is tristated in reset and when at a logic high state during I/O mode.
7. During a 0 to 1 transition, a one-shot drives the ports hard for two clock cycles. This measurement reflects port in transition mode.
8. Ports 1, 2, and 3 source transition current when being pulled down externally. It reaches its maximum at approximately 2V.
9.  $0.45 < V_{IN} < V_{CC}$ . Not a high impedance input. This port is a weak address holding latch in Bus Mode. Peak current occurs near the input transition point of the latch, approximately 2V.
10.  $0.45 < V_{IN} < V_{CC}$ .  $RST=5.5V$ . This condition mimics operation of pins in I/O mode.
11.  $V_{CC}=0V$ ,  $V_{BAT}=3.3V$ . 32.768 KHz crystal with 12.5 pF load capacitance between RTCX1 and RTCX2 pins. RTCE bit set to 1.

**TYPICAL  $I_{CC}$  VERSUS FREQUENCY** Figure 6



PARAMETER	SYMBOL	33 MHz		VARIABLE CLOCK		UNITS
		MIN	MAX	MIN	MAX	
Oscillator Frequency	$1/t_{CLCL}$	0	33	0	33	MHz
ALE Pulse Width	$t_{LHLL}$	40		$1.5t_{CLCL}-5$		ns
Port 0 Address Valid to ALE Low	$t_{AVLL}$	10		$0.5t_{CLCL}-5$		ns
Address Hold after ALE Low	$t_{LLAX1}$	10		$0.5t_{CLCL}-5$		ns
ALE Low to Valid Instruction In	$t_{LLIV}$		56		$2.5t_{CLCL}-20$	ns
ALE Low to $\overline{PSEN}$ Low	$t_{LLPL}$	10		$0.5t_{CLCL}-5$		ns
$\overline{PSEN}$ Pulse Width	$t_{PLPH}$	55		$2t_{CLCL}-5$		ns
$\overline{PSEN}$ Low to Valid Instr. In	$t_{PLIV}$		41		$2t_{CLCL}-20$	ns
Input Instruction Hold after $\overline{PSEN}$	$t_{PXIX}$	0		0		ns
Input Instruction Float after $\overline{PSEN}$	$t_{PXIZ}$		26		$t_{CLCL}-5$	ns
Port 0 Address to Valid Instr. In	$t_{AVIV}$		71		$3t_{CLCL}-20$	ns
Port 2 Address to Valid Instr. In	$t_{AVIV2}$		81		$3.5t_{CLCL}-25$	ns
$\overline{PSEN}$ Low to Address Float	$t_{PLAZ}$		0		0	ns

#### NOTES FOR AC ELECTRICAL CHARACTERISTICS:

All parameters apply to both commercial and industrial temperature range operation unless otherwise noted. All signals rated over operating temperature. All signals characterized with load capacitance of 80 pF except Port 0, ALE,  $\overline{PSEN}$ ,  $\overline{RD}$  and  $\overline{WR}$  with 100 pF. Interfacing to memory devices with float times (turn off times) over 25 ns may cause contention. This will not damage the parts, but will cause an increase in operating current.

**MOVX CHARACTERISTICS USING STRETCH MEMORY CYCLES** (0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	VARIABLE CLOCK		UNITS	STRETCH
		MIN	MAX		
Data Access ALE Pulse Width	$t_{LHLL2}$	$1.5t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Address Hold after ALE Low for MOVX Write	$t_{LLAX2}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Pulse Width	$t_{RLRH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{WR}$ Pulse Width	$t_{WLWH}$	$2t_{CLCL}-5$ $t_{MCS}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Valid Data In	$t_{RLDV}$		$2t_{CLCL}-20$ $t_{MCS}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Hold after Read	$t_{RHDZ}$	0		ns	
Data Float after Read	$t_{RHDZ}$		$t_{CLCL}-5$ $2t_{CLCL}-5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to Valid Data In	$t_{LLDV}$		$2.5t_{CLCL}-20$ $t_{MCS}+t_{CLCL}-40$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to Valid Data In	$t_{AVDV1}$		$3t_{CLCL}-20$ $t_{MCS}+1.5t_{CLCL}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to Valid Data In	$t_{AVDV2}$		$3.5t_{CLCL}-20$ $t_{MCS}+2t_{CLCL}-20$	ns	$t_{MCS}=0$ $t_{MCS}>0$
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$t_{LLWL}$	$0.5t_{CLCL}-5$ $t_{CLCL}-5$	$0.5t_{CLCL}+5$ $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 0 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL1}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Port 2 Address to $\overline{RD}$ or $\overline{WR}$ Low	$t_{AVWL2}$	$1.5t_{CLCL}-10$ $2.5t_{CLCL}-10$		ns	$t_{MCS}=0$ $t_{MCS}>0$
Data Valid to $\overline{WR}$ Transition	$t_{QVWX}$	-5		ns	
Data Hold after Write	$t_{WHQX}$	$t_{CLCL}-5$ $2t_{CLCL}-5$		ns	$t_{MCS}=0$ $t_{MCS}>0$
$\overline{RD}$ Low to Address Float	$t_{RLAZ}$		$-0.5t_{CLCL}-5$	ns	
$\overline{RD}$ or $\overline{WR}$ High to ALE High	$t_{WHLH}$	0 $t_{CLCL}-5$	10 $t_{CLCL}+5$	ns	$t_{MCS}=0$ $t_{MCS}>0$

NOTE:  $t_{MCS}$  is a time period related to the Stretch memory cycle selection. The following table shows the value of  $t_{MCS}$  for each Stretch selection.

M2	M1	M0	MOVX CYCLES	$t_{MCS}$
0	0	0	2 machine cycles	0
0	0	1	3 machine cycles (default)	$4 t_{CLCL}$
0	1	0	4 machine cycles	$8 t_{CLCL}$
0	1	1	5 machine cycles	$12 t_{CLCL}$
1	0	0	6 machine cycles	$16 t_{CLCL}$
1	0	1	7 machine cycles	$20 t_{CLCL}$
1	1	0	8 machine cycles	$24 t_{CLCL}$
1	1	1	9 machine cycles	$28 t_{CLCL}$

**EXTERNAL CLOCK CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Clock High Time	$t_{CHCX}$	10			ns	
Clock Low Time	$t_{CLCX}$	10			ns	
Clock Rise Time	$t_{CLCL}$			5	ns	
Clock Fall Time	$t_{CHCL}$			5	ns	

**SERIAL PORT MODE 0 TIMING CHARACTERISTICS**(0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Serial Port Clock Cycle Time SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XLXL}$		$12t_{CLCL}$ $4t_{CLCL}$		ns ns	
Output Data Setup to Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{QVXH}$		$10t_{CLCL}$ $3t_{CLCL}$		ns ns	
Output Data Hold from Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHGX}$		$2t_{CLCL}$ $t_{CLCL}$		ns ns	
Input Data Hold after Clock Rising SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHDX}$		$t_{CLCL}$ $t_{CLCL}$		ns ns	
Clock Rising Edge to Input Data Valid SM2=0, 12 clocks per cycle SM2=1, 4 clocks per cycle	$t_{XHDV}$		$11t_{CLCL}$ $3t_{CLCL}$		ns ns	

**EXPLANATION OF AC SYMBOLS**

In an effort to remain compatible with the original 8051 family, this device specifies the same parameters as such devices, using the same symbols. For completeness, the following is an explanation of the symbols.

t	Time
A	Address
C	Clock
D	Input data
H	Logic level high
L	Logic level low
I	Instruction
P	PSEN
Q	Output data
R	RD signal
V	Valid
W	WR signal
X	No longer a valid logic level
Z	Tristate

**POWER CYCLE TIMING CHARACTERISTICS** (0°C to 70°C;  $V_{CC}=4.0V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Cycle Start-up Time	$t_{CSU}$		1.8		ms	1
Power-on Reset Delay	$t_{POR}$			65536	$t_{CLCL}$	2

**NOTES FOR POWER CYCLE TIMING:**

1. Start-up time for crystals varies with load capacitance and manufacturer. Time shown is for an 11.0592 MHz crystal manufactured by Fox.
2. Reset delay is a synchronous counter of crystal oscillations after crystal start-up. At 33 MHz, this time is 1.99 ms.

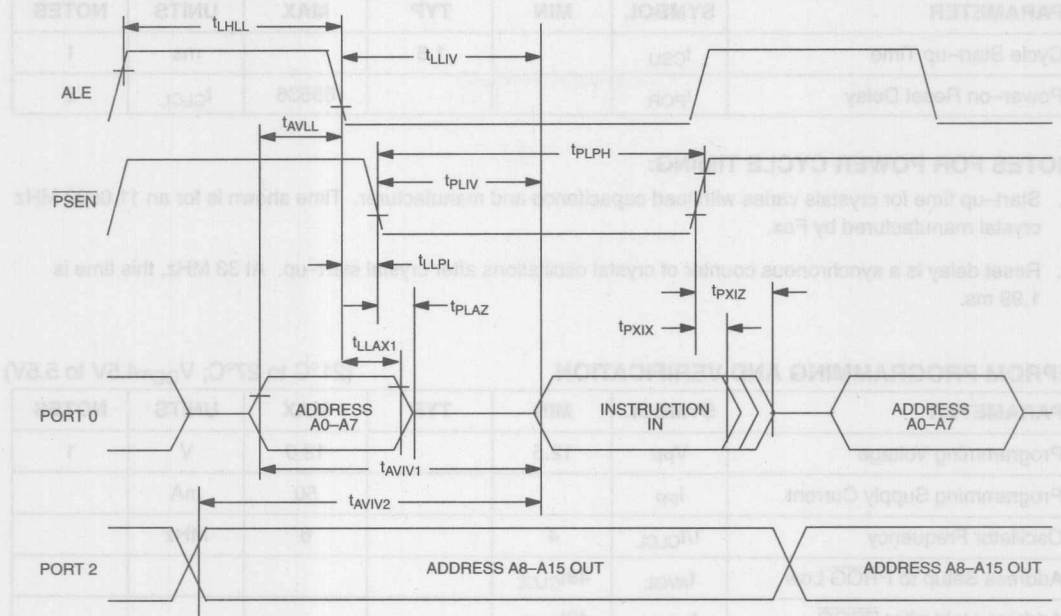
**EPROM PROGRAMMING AND VERIFICATION** (21°C to 27°C;  $V_{CC}=4.5V$  to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Programming Voltage	$V_{PP}$	12.5		13.0	V	1
Programming Supply Current	$I_{PP}$			50	mA	
Oscillator Frequency	$1/t_{CLCL}$	4		6	MHz	
Address Setup to $\overline{PROG}$ Low	$t_{AVGL}$	$48t_{CLCL}$				
Address Hold after $\overline{PROG}$	$t_{GHAX}$	$48t_{CLCL}$				
Data Setup to $\overline{PROG}$ Low	$t_{DVGL}$	$48t_{CLCL}$				
Data Hold after $\overline{PROG}$	$t_{GHDX}$	$48t_{CLCL}$				
Enable High to $V_{PP}$	$t_{EHS}$	$48t_{CLCL}$				
$V_{PP}$ Setup to $\overline{PROG}$ Low	$t_{SHGL}$	10			$\mu s$	
$V_{PP}$ Hold after $\overline{PROG}$	$t_{GHSL}$	10			$\mu s$	
$\overline{PROG}$ Width	$t_{GLGH}$	90		110	$\mu s$	
Address to Data Valid	$t_{AVQV}$			$48t_{CLCL}$		
Enable Low to Data Valid	$t_{ELQV}$			$48t_{CLCL}$		
Data Float after Enable	$t_{EHQZ}$	0		$48t_{CLCL}$		
$\overline{PROG}$ High to $\overline{PROG}$ Low	$t_{GHGL}$	10			$\mu s$	

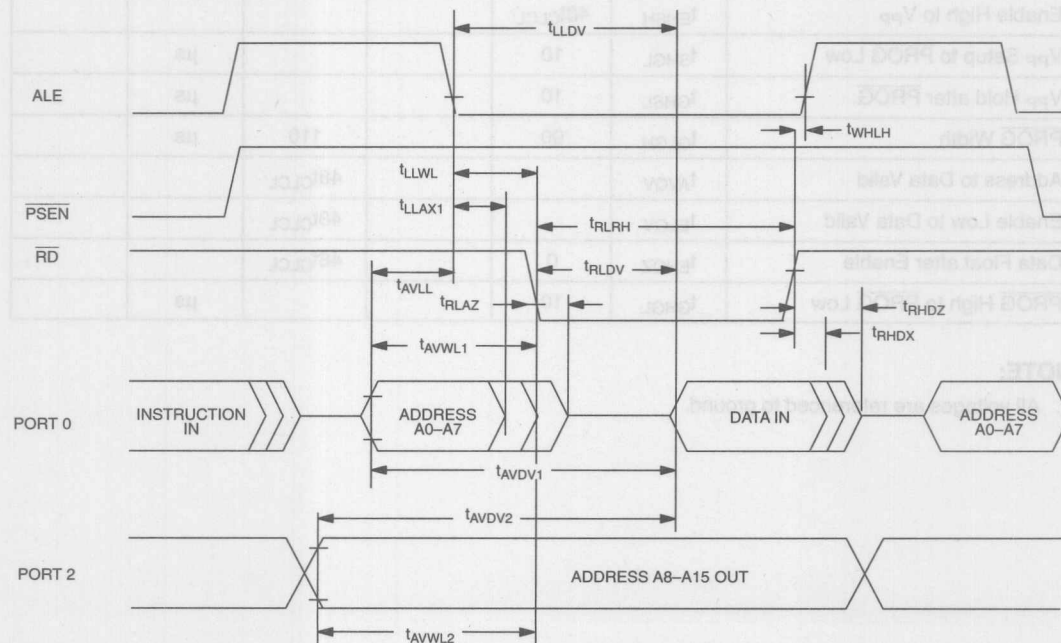
**NOTE:**

1. All voltages are referenced to ground.

## EXTERNAL PROGRAM MEMORY READ CYCLE

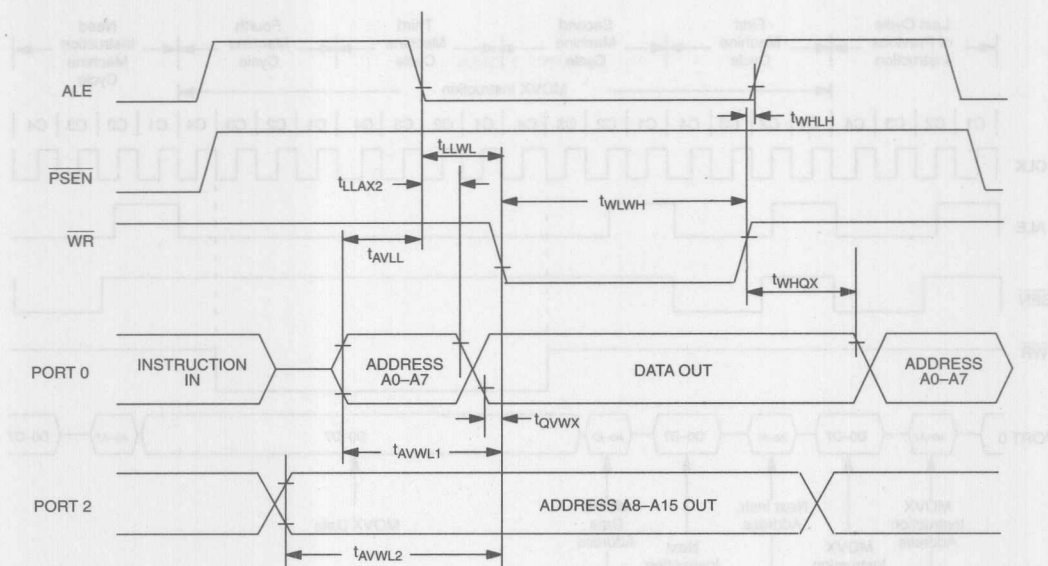


## EXTERNAL DATA MEMORY READ CYCLE

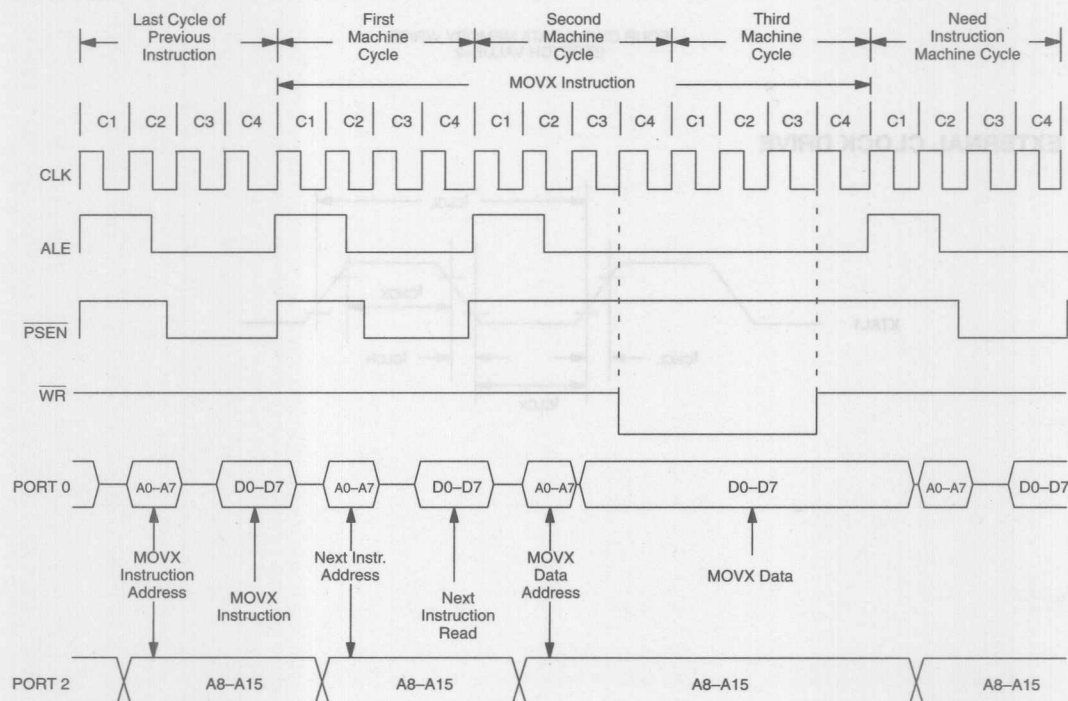




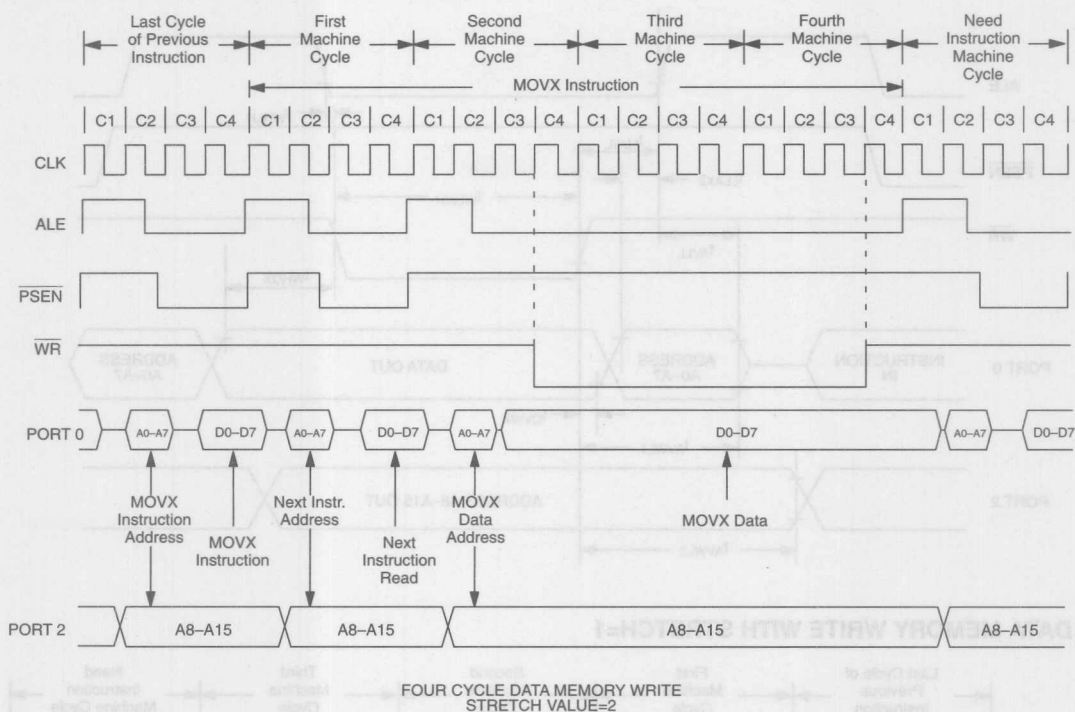
## DATA MEMORY WRITE CYCLE



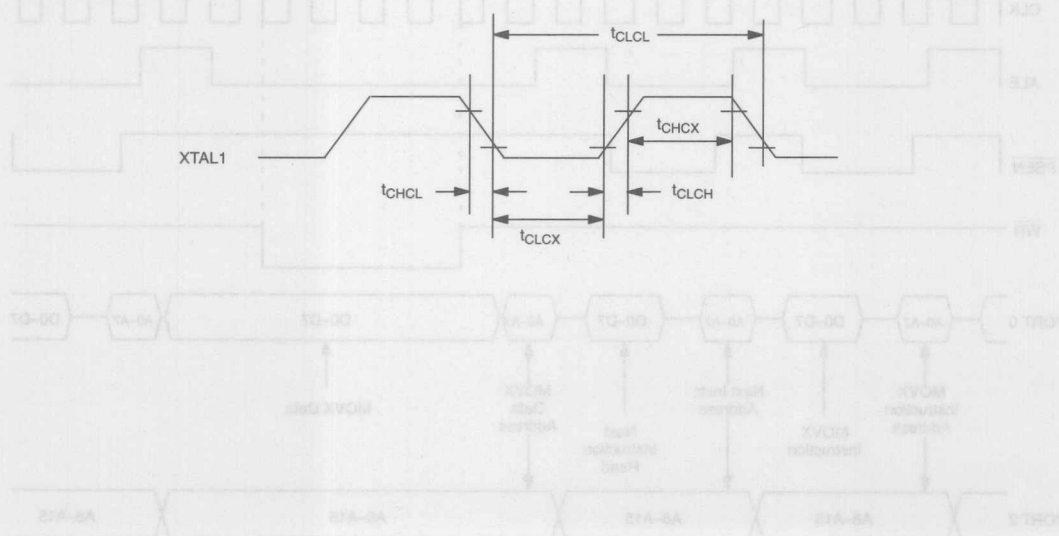
## DATA MEMORY WRITE WITH STRETCH=1



## DATA MEMORY WRITE WITH STRETCH=2

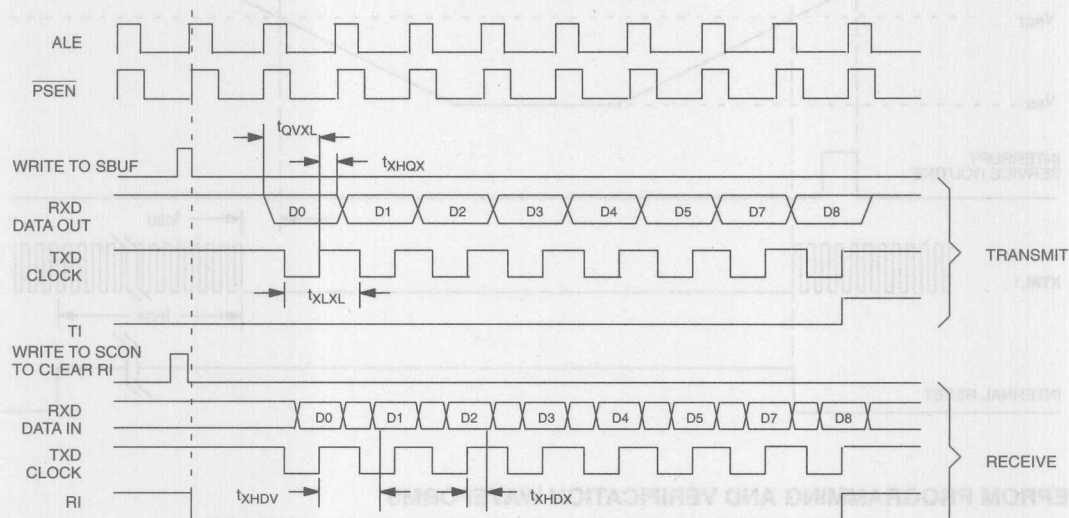


## EXTERNAL CLOCK DRIVE

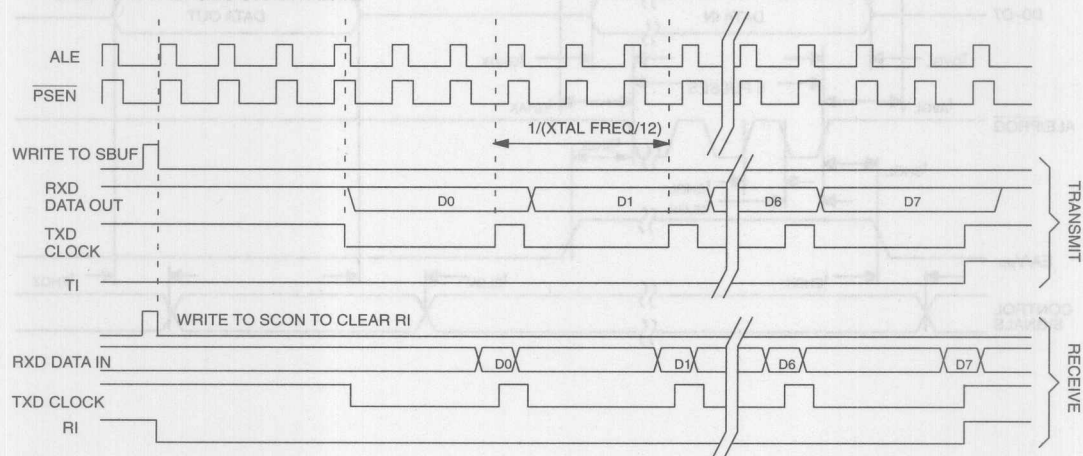


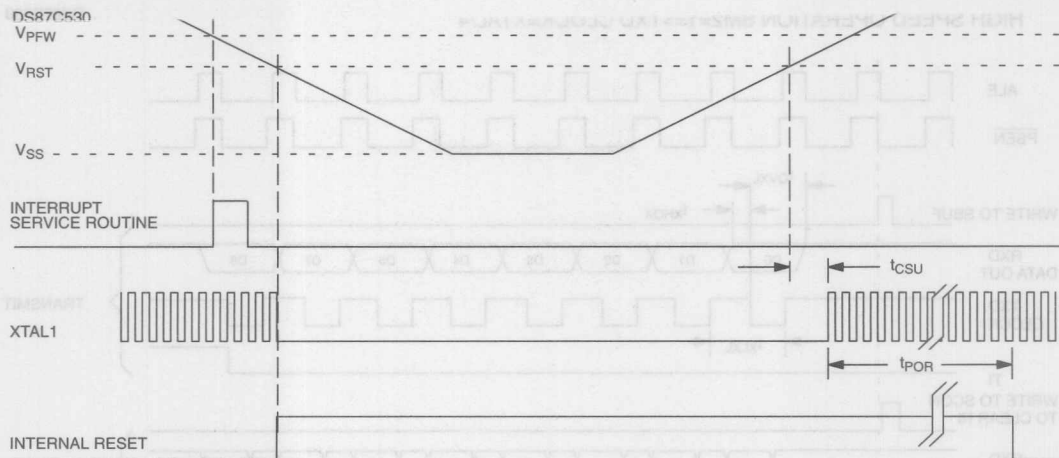
# SERIAL PORT MODE 0 TIMING

SERIAL PORT 0 (SYNCHRONOUS MODE)  
HIGH SPEED OPERATION  $SM2=1 \Rightarrow TXD\ CLOCK=XTAL/4$

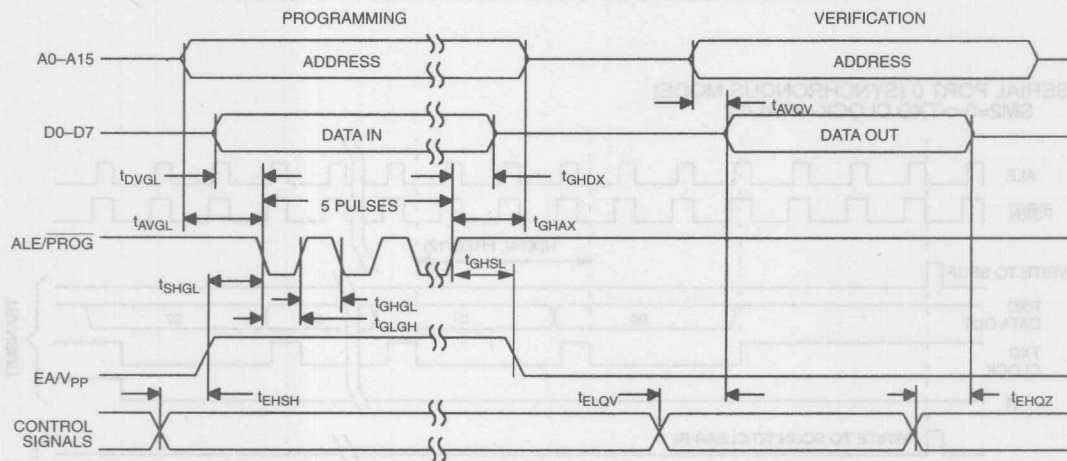


SERIAL PORT 0 (SYNCHRONOUS MODE)  
 $SM2=0 \Rightarrow TXD\ CLOCK=XTAL/12$





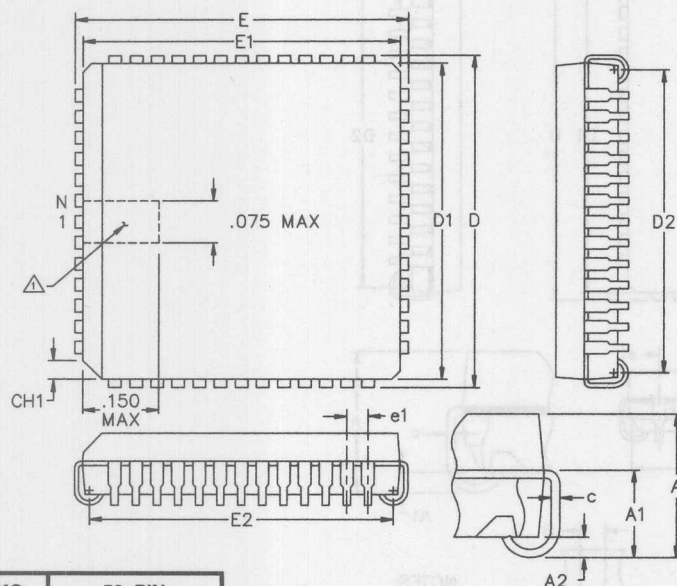
## EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



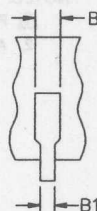
## 52-PIN PLCC

NOTE:

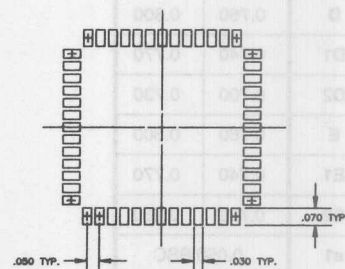
- △ PIN-1 IDENTIFIER TO BE LOCATED IN ZONE INDICATED.  
 2. CONTROLLING DIMENSIONS ARE IN INCHES.



PKG	52-PIN	
DIM	MIN	MAX
A	0.165	0.180
A1	0.090	0.120
A2	0.020	—
B	0.026	0.032
B1	0.013	0.021
c	0.008	0.013
CH1	0.042	0.048
D	0.785	0.795
D1	0.750	0.756
D2	0.690	0.730
E	0.785	0.795
E1	0.750	0.756
E2	0.690	0.730
e1	0.050 BSC	
N	52	—



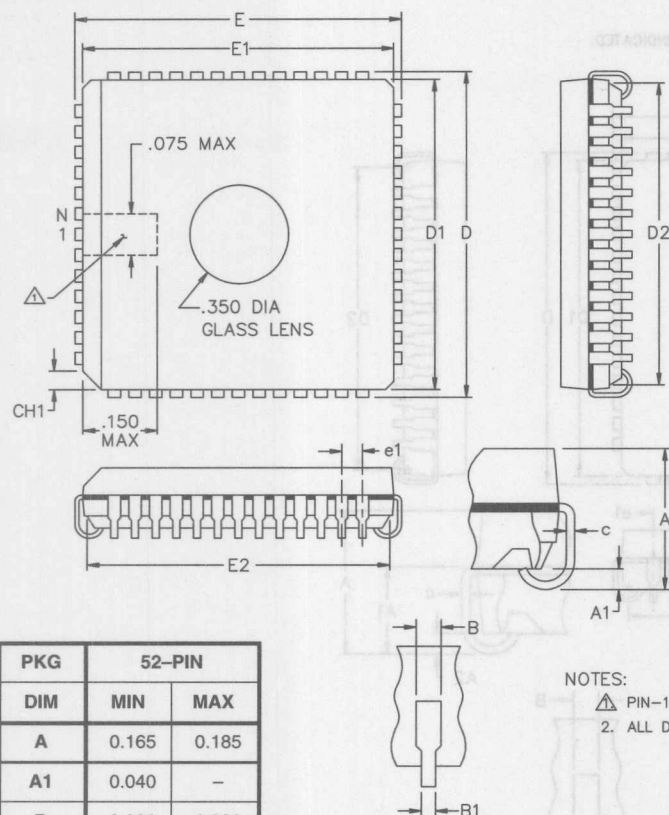
52 PIN QUAD (PLCC)  
 SUGGESTED PAD LAYOUT



56-G4006-001

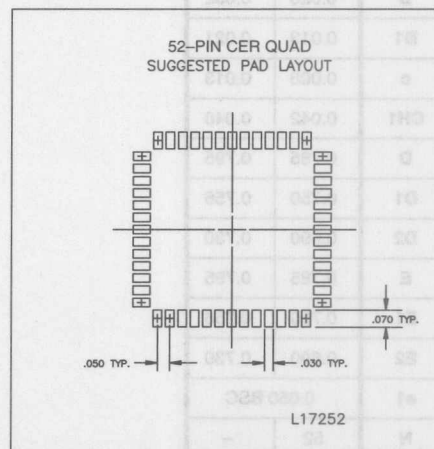


## 52-PIN CER QUAD



PKG	52-PIN	
DIM	MIN	MAX
A	0.165	0.185
A1	0.040	—
B	0.026	0.032
B1	0.013	0.021
c	0.008	0.013
CH1-45°	0.035	0.040
D	0.760	0.800
D1	0.740	0.770
D2	0.700	0.730
E	0.760	0.800
E1	0.740	0.770
E2	0.700	0.730
e1	0.050 BSC	
N	52	—

56-G4007-001



# DALLAS

SEMICONDUCTOR

## DS80C323

### Low-Power Micro

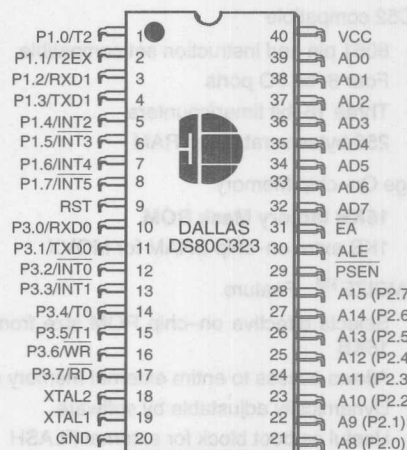
#### FEATURES

- 80C32-Compatible
  - Pin-compatible
  - Standard 8051 instruction set
  - Four 8-bit I/O ports
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
  - Multiplexed address/data bus
  - Addresses 64KB ROM and 64KB RAM
- Operates between 2.7V and 5.5V
- High-speed architecture
  - 4 clocks/machine cycle (8032=12)
  - Wasted cycles removed
  - Runs DC to 20 MHz clock rate @ 2.7V
  - Single-cycle instruction in 200 ns
  - Uses less power for equivalent work
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM /peripherals
- High integration controller includes:
  - Power-fail reset
  - Programmable Watchdog timer
  - Early-warning power-fail interrupt
- Two full-duplex hardware serial ports
- 13 total interrupt sources with six external
- Available in 40-pin DIP, 44-pin PLCC and TQFP

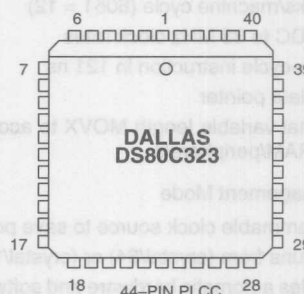
#### DESCRIPTION

The DS80C323 is a fast 80C31/80C32-compatible microcontroller. Wasted clock and memory cycles have been removed using a redesigned processor core. As a result, every 8051 instruction is executed between 1.5 and 3 times faster than the original for the same crystal speed. Typical applications will see a speed improvement of 2.5 times using the same code and same crystal. At 2.7V, the DS80C323 offers a maximum crystal rate of 20 MHz, resulting in apparent execution speeds of 50 MHz (approximately 2.5X).

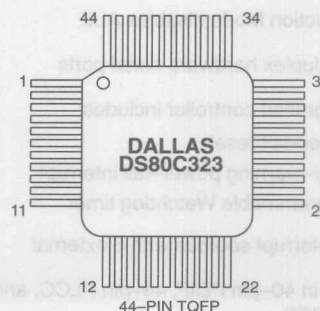
#### PIN ASSIGNMENT



40-PIN DIP



44-PIN PLCC



44-PIN TQFP

# DALLAS

SEMICONDUCTOR

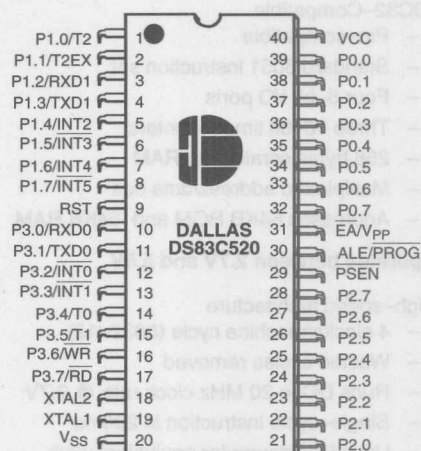
## DS83C520

### ROM High-Speed Micro

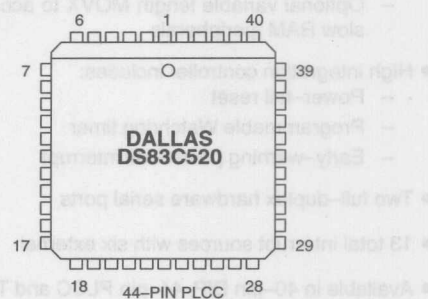
#### FEATURES

- 80C52 compatible
  - 8051 pin and instruction set compatible
  - Four 8-bit I/O ports
  - Three 16-bit timer/counters
  - 256 bytes scratchpad RAM
- Large On-chip Memory
  - **16KB factory Mask ROM**
  - 1KB extra on-chip SRAM for MOVX
- ROMSIZE™ Feature
  - Selects effective on-chip ROM size from 0 to 16KB
  - Allows access to entire external memory map
  - Dynamically adjustable by software
  - Useful as boot block for external FLASH
- High-Speed Architecture
  - 4 clocks/machine cycle (8051 = 12)
  - Runs DC to 33 MHz clock rates
  - Single-cycle instruction in 121 ns
  - Dual data pointer
  - Optional variable length MOVX to access fast/slow RAM/peripherals
- Power Management Mode
  - Programmable clock source to save power
  - CPU runs from (crystal/64) or (crystal/1024)
  - Provides automatic hardware and software exit
- EMI Reduction Mode disables ALE
- Two full-duplex hardware serial ports
- High integration controller includes:
  - Power-fail reset
  - Early-warning power-fail interrupt
  - Programmable Watchdog timer
- 13 total interrupt sources with 6 external
- Available in 40-pin PDIP, 44-pin PLCC, and 44-pin TQFP

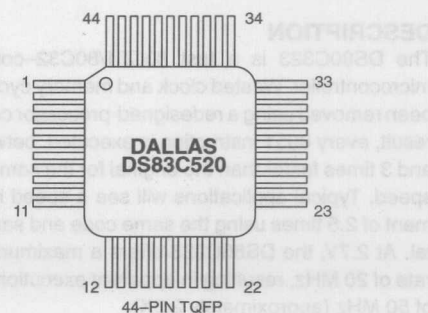
#### PACKAGE OUTLINE



40-PIN PDIP



44-PIN PLCC



44-PIN TQFP

## APPLICATION NOTES

SEMICONDUCTOR

Application Note 26  
The D880C320 as a Drop-In  
Replacement for the 8032

something that must be considered, and may be important in some systems.

## Software Issues

The other issue having to do with speed considerations is the use of software timing. It is frequently the case that software writers will use the presumed constant execution speed of a processor as a real time reference. Often a tight loop that requires a known number of clocks to execute will be used for generating delays. Since the D880C320 executes instructions much more quickly than the standard 8032, these previously designed timing loops will no longer produce the originally intended results. While using software timing loops is generally accepted as undesirable software design, in practice they are used rather frequently in embedded applications. The D880C320 was designed so that the internal timer default to a condition where they behave exactly as the timers in the 8032. If application code is written to take use of these timers rather than software delays, the code will run as originally intended.

## POWER-ON RESET

The D880C320 incorporates circuitry to generate its own power-on reset function. While the RST pin may still be connected to an external reset generating circuit, this on-board feature is provided as a convenience to new designs. The fact that the processor has its own reset function is a benefit in most cases; however, there are situations where the on-board reset is not exactly what the user wants. One could conceive of situations where the reset may not be at exactly the desired voltage level or last for exactly the desired duration. One example of this is could be the case where battery backed RAM is used for storage. If the RAM contains its own voltage detection circuitry and does not become unpowered at the same voltage as the D880C320 leaves reset (4.0 volts), then the processor could be accessing protected RAM. While these cases are not

## INTRODUCTION

The D880C320 high speed micro is another member of Dallas Semiconductor's 8032 instruction set compatible family of microprocessors. It was designed with the same pinout and basic resources as a conventional 8032, but has significantly enhanced performance capabilities and a number of additional resources. Since the instruction set and pinout are the same, many situations allow it to be used as a drop-in replacement. When doing so, however, there are some issues that must be taken into account. This application note discusses those issues.

## PROCESSOR SPEED

While the D880C320 is 100 percent compatible with the 8032 instruction set, the execution of the instructions has been streamlined for increased performance. A single byte instruction that previously required 12 clocks to complete now executes in 4 clocks. In addition, the D880C320 can accept clocks up to 25MHz where some versions of the 8032 the maximum was 12MHz. Because of this higher performance, there are issues relating to processor speed that must be considered when evaluating the D880C320 as a drop-in replacement for the 8032.

## Memory Interface

Since the basic instruction execution time has been streamlined in the D880C320, the time available to transfer data to and from memory has also been reduced. This means that for the same frequency cycle, there is less time available for memory access. A simple example illustrates this point. The data sheet for the 8032 stipulates that when using a 12MHz crystal, the program memory must have an address access time of 302 ns or less (neglecting any address latch overhead). A D880C320 also using a 12MHz crystal requires a memory with an address access time of 130 ns or less. While this is not a tremendous difference, it

1. Details on selecting the correct speed memory devices for the D880C320 may be found in Dallas Semiconductor's Application Note 27 entitled "D880C320 Memory Interface Timing".

2. Data for the 8032 from the Intel "7-bit Embedded Controller" data book dated 1981.

# DALLAS

## SEMICONDUCTOR

## Application Note 56

### The DS80C320 as a Drop-In Replacement for the 8032

#### INTRODUCTION

The DS80C320 high speed micro is another member of Dallas Semiconductor's 8051 instruction set compatible family of microprocessors. It was designed with the same pinout and basic resources as a conventional 8032, but has significantly enhanced performance capabilities and a number of additional resources. Since the instruction set and pinout are the same, many situations allow it to be used as a drop-in replacement. When doing so however, there are some issues that must be taken into account. This application note discusses those issues.

#### PROCESSOR SPEED

While the DS80C320 is 100 percent compatible with the 8051 instruction set, the execution of the instructions has been streamlined for increased performance. A single byte instruction that previously required 12 clocks to complete now executes in 4 clocks. In addition, the DS80C320 can accept clocks up to 25MHz where in some versions of the 8032 the maximum was 12MHz. Because of this higher performance, there are issues relating to processor speed that must be considered when evaluating the DS80C320 as a drop-in replacement for the 8032.

#### Memory Interface<sup>1</sup>

Since the basic instruction execution time has been streamlined in the DS80C320, the time available to transfer data to and from memory has also been reduced. This means that for the same frequency crystal, there is less time available for memory access. A simple example illustrates this point. The data sheet for the 8032<sup>2</sup> stipulates that, when using a 12 MHz crystal, the program memory must have an address access time of 302 ns or less (neglecting any address latch overhead). A DS80C320 also using a 12MHz crystal requires a memory with an address access time of 230 ns or less. While this is not a tremendous difference, it is

something that must be considered, and may be important in some systems.

#### Software Loops

The other issue having to do with speed considerations is the use of software timing. It is frequently the case that software writers will use the presumed constant execution speed of a processor as a real time reference. Often a tight loop that requires a known number of clocks to execute will be used for generating delays. Since the DS80C320 executes instructions much more quickly than the standard 8032, these previously designed timing loops will no longer produce the originally intended results. While using software timing loops is generally accepted as undesirable software design, in practice they are used rather frequently in embedded applications. The DS80C320 was designed so that the internal timers default to a condition where they behave exactly as the timers in the 8032. If application code is written to make use of these timers rather than software delays, the code will run as originally intended.

#### POWER-ON RESET

The DS80C320 incorporates circuitry to generate its own power-on reset function. While the RST pin may still be connected to an external reset generating circuit, this on-board feature is provided as a convenience to new designs. The fact that the processor has its own reset function is a benefit in most cases, however, there are situations where the on-board reset is not exactly what the user wants. One could conceive of situations where the reset may not be at exactly the desired voltage level or last for exactly the desired duration. One example of this is could be the case where battery backed RAM is used for storage. If the RAM contains its own voltage detection circuitry and does not become unprotected at the same voltage as the DS80C320 leaves reset (4.0 volts), then the processor could be accessing protected RAM. While these cases are not

1. Details on selecting the correct speed memory devices for the DS80C320 may be found in Dallas Semiconductor's Application Note 57 entitled "DS80C320 Memory Interface Timing".

2. Data for the 8032 from the Intel "8-Bit Embedded Controllers" data book dated 1991.



common, they remain something to consider for each specific application.

### POWER CONSUMPTION

In addition to being a higher performance device, the DS80C320 is also a lower power device than the 8032 when equivalent work is considered. All CMOS parts exhibit the property that they consume more power as

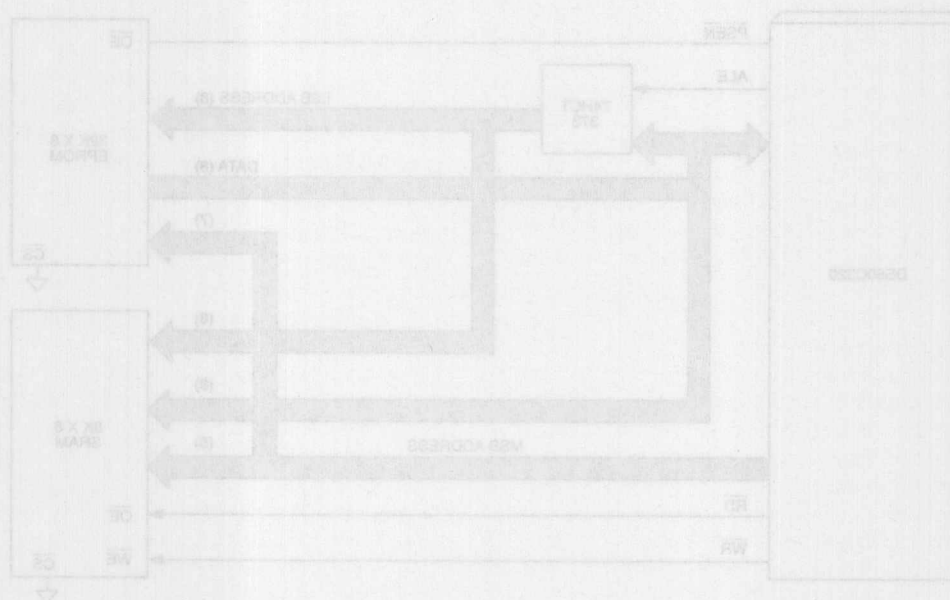
their speed goes up. Since the DS80C320 is a higher speed part, it will consume more power for a given crystal frequency. However, if an equivalent amount of work is considered, it consumes slightly less power than a conventional 8032. This difference in power consumption is probably only important for battery powered applications, in which case stop mode power is likely to be more important.

### INTRODUCTION

Dallas Semiconductor's DS80C320 processor provides extensive new application opportunities due to its increased throughput. The increased speed, however, also requires attention to the timing requirements of the memory that interfaces with the processor. This application note identifies the critical timing paths associated with the memory interface and identifies memory speeds required for various CPU crystal frequencies.

A typical configuration for a DS80C320 processor system is shown in Figure 1. A 32K x 8 EPROM is used to

A TYPICAL DS80C320 SYSTEM CONFIGURATION Figure 1

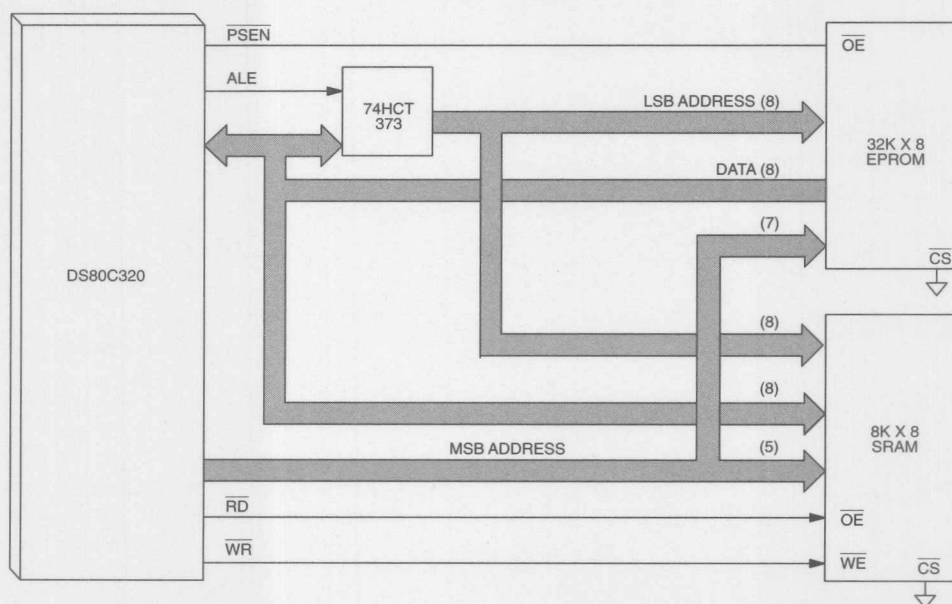


## INTRODUCTION

A typical configuration for a DS80C320 processor system is shown in Figure 1. A 32K x 8 EPROM is used to

hold program information, and a 8K x 8 Static RAM is used for data storage. The Least Significant Byte (LSB) of the memories' address is time multiplexed with data on processor pins AD7 through AD0. The signal ALE from the processor goes high before a new address is placed on the bus, and goes low before it is removed. In Figure 1, the action of ALE going low is used to latch the address into a 74HCT373 8-bit transparent latch. The 373 then provides its latched address output to the memories while the AD7 through AD0 CPU bus carries data. The MSB of the address is not multiplexed, and is available on port pins P2.7 through P2.0 (A15-A9).

### A TYPICAL DS80C320 SYSTEM CONFIGURATION Figure 1

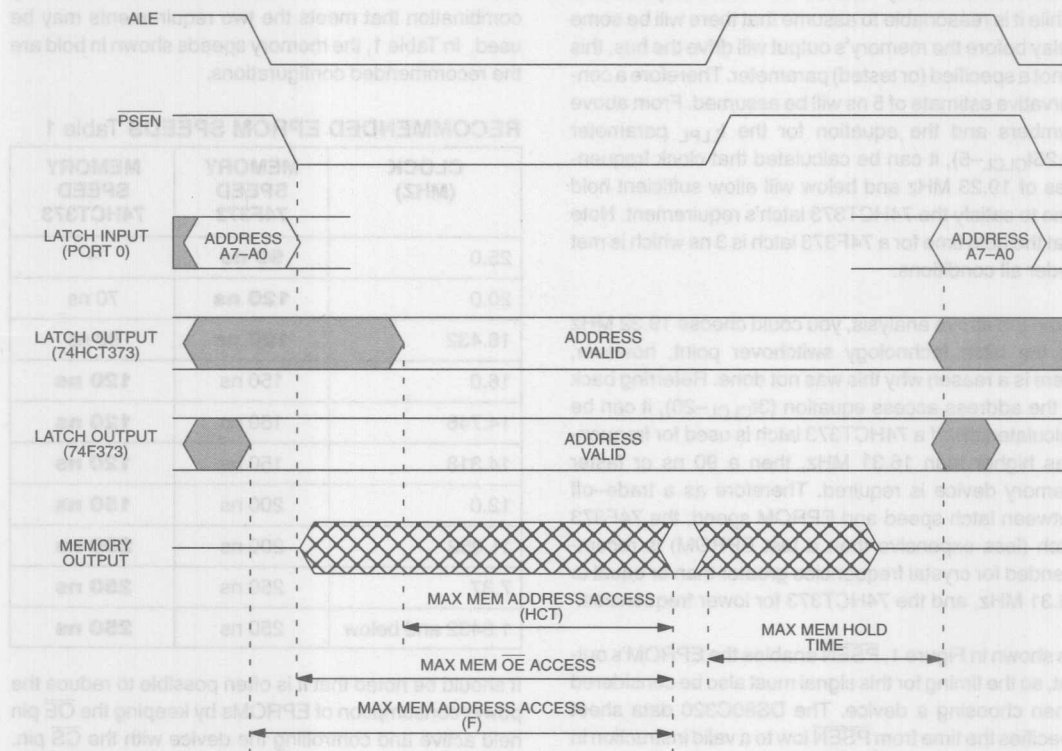


## PROGRAM MEMORY

Some of the signals used in interfacing the DS80C320 with EPROM Program memory are shown in Figure 2. As can be seen, timing relationships for two latch technologies (HCT, and F) are shown. The technology selected for this latch is critical to the memory selection. The 74HCT373 has a worst case propagation delay from input to output (D to Q) of 44 ns<sup>1</sup> while the 74F373's is 8 ns<sup>2</sup>. This results in significantly different memory

address access timing requirements depending on the family used. Examining timing parameters from the DS80C320 data sheet reveals that an instruction must be ready to read into the processor within 100 ns (parameter  $t_{AVIV1}=3t_{CLCL}-20$ )<sup>3</sup>, assuming a 25 MHz clock<sup>4</sup>. If the 44 ns propagation delay through the HCT latch is subtracted from this, you arrive at a required address access time of 56 ns.

PROGRAM MEMORY INTERFACE TIMING Figure 2



While EPROM devices with access times of 56 ns or less may be available, they are likely to be expensive. A simple and more cost effective approach to solving this timing constraint is to use a faster latch technology; an 74F373 for instance. Using the same analysis as above,

if the propagation delay of the F373 latch, 8 ns, is subtracted from the  $t_{AVIV1}$  parameter (100 ns) you arrive at an address access requirement of 92 ns. This is much easier to realize than 56 ns.

1. HCT timing information from the 1988 Texas Instruments "High Speed CMOS Logic" data book.
2. F timing information from the Signetics "Fast Data Manual 1986" data book.
3. For details on the equations presented in this document, refer to the DS80C320 data sheet.
4.  $t_{CLCL}$  is the period of the crystal frequency, and is equal to 40 ns for a 25 MHz crystal.

There is another timing constraint that suggests the use of an "F" type part in faster applications. On a 74HCT373 latch, the minimum required hold time of the input after the latch enable (ALE) goes low may be as much as 13 ns. The latch's input, the address out of the processor, is held until it is driven by the memory's output. This output is enabled by  $\overline{\text{PSEN}}$ . Again referring to the data sheet, it can be seen that  $\overline{\text{PSEN}}$  may occur as quickly as 5 ns after ALE falls (parameter  $t_{\text{LLPL}}$ ). If the memory's output begins to drive the bus immediately after  $\overline{\text{PSEN}}$  enables it, then there may only be 5 ns hold time. That obviously violates the latch's requirement. While it is reasonable to assume that there will be some delay before the memory's output will drive the bus, this is not a specified (or tested) parameter. Therefore a conservative estimate of 5 ns will be assumed. From above numbers and the equation for the  $t_{\text{LLPL}}$  parameter ( $0.25t_{\text{CLCL}}-5$ ), it can be calculated that clock frequencies of 19.23 MHz and below will allow sufficient hold time to satisfy the 74HCT373 latch's requirement. Note that the hold time for a 74F373 latch is 3 ns which is met under all conditions.

From the above analysis, you could choose 19.32 MHz as the latch technology switchover point, however, there is a reason why this was not done. Referring back to the address access equation ( $3t_{\text{CLCL}}-20$ ), it can be calculated that if a 74HCT373 latch is used for frequencies higher than 16.31 MHz, then a 90 ns or faster memory device is required. Therefore as a trade-off between latch speed and EPROM speed, the 74F373 latch (less expensive than a fast EPROM) is recommended for crystal frequencies greater than or equal to 16.31 MHz, and the 74HCT373 for lower frequencies.

As shown in Figure 1,  $\overline{\text{PSEN}}$  enables the EPROM's output, so the timing for this signal must also be considered when choosing a device. The DS80C320 data sheet specifies the time from  $\overline{\text{PSEN}}$  low to a valid instruction in must be no more than 70 ns (parameter  $t_{\text{PLIV}}$ ). This is therefore the maximum allowed access time from the memory's  $\overline{\text{OE}}$  pin. In summary, the two timing requirements for the selected EPROM are that the address access time must be less than 92 ns and the  $\overline{\text{OE}}$  access time must be less than 70 ns. When looking at EPROM data sheets, it can be seen that common access time combinations (address access, OE access) are 55,35;

70,40; 90,40; 120,50; 150,65; 200,75; and 250,100 ns. The 90,40 device meets both timing requirements for a 25 MHz clock on the DS80C320 and is the recommended selection.

Table 1 shows the slowest EPROM memory speeds recommended for various processor clock frequencies. If for some reason it is desirable to use devices faster than those recommended, this is possible. For this document, the memory speeds available (maximum access times from address and  $\overline{\text{OE}}$  respectively) are assumed to be those indicated above, however, any combination that meets the two requirements may be used. In Table 1, the memory speeds shown in bold are the recommended configurations.

RECOMMENDED EPROM SPEEDS Table 1

CLOCK (MHZ)	MEMORY SPEED 74F373	MEMORY SPEED 74HCT373
25.0	<b>90 ns</b>	—
20.0	<b>120 ns</b>	70 ns
18.432	<b>120 ns</b>	90 ns
16.0	150 ns	<b>120 ns</b>
14.746	150 ns	<b>120 ns</b>
14.318	150 ns	<b>120 ns</b>
12.0	200 ns	<b>150 ns</b>
11.059	200 ns	<b>200 ns</b>
7.37	250 ns	<b>250 ns</b>
1.8432 and below	250 ns	<b>250 ns</b>

It should be noted that it is often possible to reduce the power consumption of EPROMs by keeping the  $\overline{\text{OE}}$  pin held active and controlling the device with the  $\overline{\text{CS}}$  pin. When doing this, however, the access time from  $\overline{\text{CS}}$  must be more closely considered. The  $\overline{\text{CS}}$  access time is often nearly the same value as the address access time (i.e., much slower than  $\overline{\text{OE}}$  access). If power consumption is the prime system consideration, a faster device may be selected, and the  $\overline{\text{CS}}$  used to select the chip.

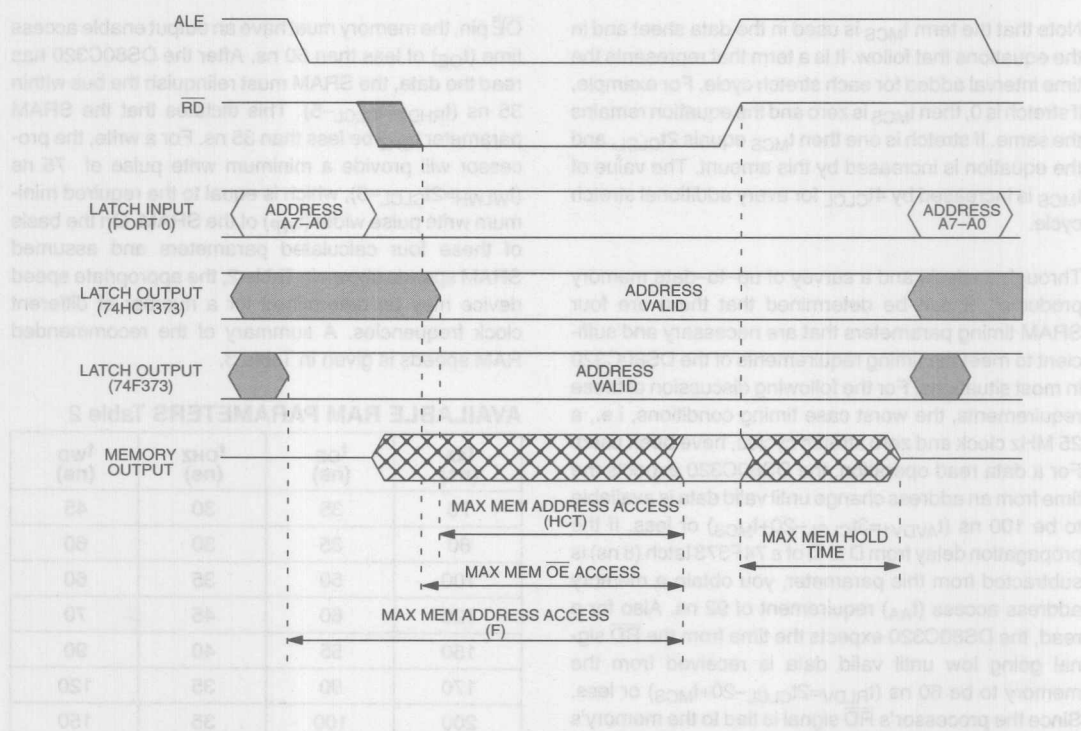
## DATA MEMORY

Choosing a data memory (RAM) device to interface to the DS80C320 is much easier than choosing an EPROM device because of the flexibility designed into the processor. The DS80C320 provides a unique feature that allows the application software to adjust the speed at which data memory is accessed. The processor is capable of performing a MOVX instruction in as little as two instruction cycles (eight oscillator clocks). However, this value can be "stretched" as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic. On power up, the DS80C320 defaults to a stretch value of one resulting in a three cycle MOVX instruction. This default condition is a convenience to existing designs that may not have fast RAM in place. For the user who needs maximum performance, a stretch value of zero may be selected by software, resulting in a two machine cycle MOVX instruction. Even in high speed systems, it may not be

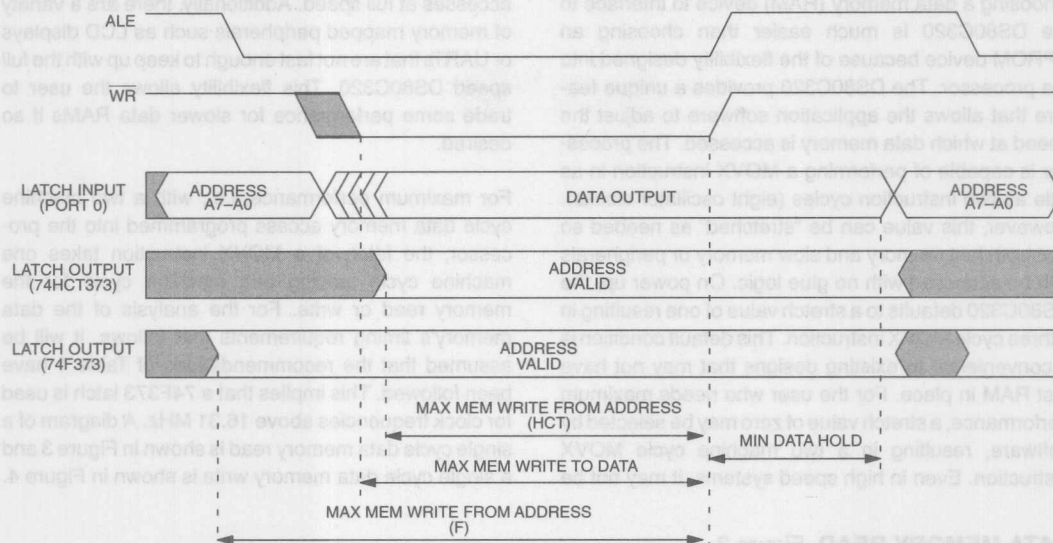
necessary or desirable to perform data memory accesses at full speed. Additionally, there are a variety of memory mapped peripherals such as LCD displays or UARTs that are not fast enough to keep up with the full speed DS80C320. This flexibility allows the user to trade some performance for slower data RAMs if so desired.

For maximum performance, i.e., with a two machine cycle data memory access programmed into the processor, the fetch of a MOVX instruction takes one machine cycle leaving one machine cycle for the memory read or write. For the analysis of the data memory's timing requirements that follows, it will be assumed that the recommendations of Table 1 have been followed. This implies that a 74F373 latch is used for clock frequencies above 16.31 MHz. A diagram of a single cycle data memory read is shown in Figure 3 and a single cycle data memory write is shown in Figure 4.

DATA MEMORY READ Figure 3





**DATA MEMORY WRITE Figure 4**

Note that the term  $t_{MCS}$  is used in the data sheet and in the equations that follow. It is a term that represents the time interval added for each stretch cycle. For example, if stretch is 0, then  $t_{MCS}$  is zero and the equation remains the same. If stretch is one then  $t_{MCS}$  equals  $2t_{CLCL}$ , and the equation is increased by this amount. The value of  $t_{MCS}$  is increased by  $4t_{CLCL}$  for every additional stretch cycle.

Through analysis and a survey of up-to-date memory products<sup>5</sup>, it can be determined that there are four SRAM timing parameters that are necessary and sufficient to meet the timing requirements of the DS80C320 in most situations. For the following discussion of these requirements, the worst case timing conditions, i.e., a 25 MHz clock and zero stretch cycles, have been used. For a data read operation, the DS80C320 expects the time from an address change until valid data is available to be 100 ns ( $t_{AVDV1}=3t_{CLCL}-20+t_{MCS}$ ) or less. If the propagation delay from D to Q of a 74F373 latch (8 ns) is subtracted from this parameter, you obtain a memory address access ( $t_{AA}$ ) requirement of 92 ns. Also for a read, the DS80C320 expects the time from the  $\overline{RD}$  signal going low until valid data is received from the memory to be 60 ns ( $t_{RLDV}=2t_{CLCL}-20+t_{MCS}$ ) or less. Since the processor's  $\overline{RD}$  signal is tied to the memory's

$\overline{OE}$  pin, the memory must have an output enable access time ( $t_{OE}$ ) of less than 60 ns. After the DS80C320 has read the data, the SRAM must relinquish the bus within 35 ns ( $t_{RHDZ}=t_{CLCL}-5$ ). This dictates that the SRAM parameter  $t_{OHZ}$  be less than 35 ns. For a write, the processor will provide a minimum write pulse of 75 ns ( $t_{WLWH}=2t_{CLCL}-5$ ), which is equal to the required minimum write pulse width ( $t_{WP}$ ) of the SRAM. On the basis of these four calculated parameters and assumed SRAM speeds shown in Table 2, the appropriate speed device may be determined for a number of different clock frequencies. A summary of the recommended RAM speeds is given in Table 3.

**AVAILABLE RAM PARAMETERS Table 2**

$t_{AA}$ (ns)	$t_{OE}$ (ns)	$t_{OHZ}$ (ns)	$t_{WD}$ (ns)
70	35	30	45
80	35	30	60
100	50	35	60
120	60	45	70
150	55	40	90
170	80	35	120
200	100	35	150

5. Memory data books from Dallas Semiconductor 1992–1993, Fujitsu 1990, Hitachi #M18, Micron 1992, Mosel 1991–1992, NEC 1989, Sony 1991 were surveyed for suitable RAM products.

Table 3 illustrates the point that even with a 25 MHz clock, relatively slow SRAM devices may be selected if a single stretch cycle (default condition) is used. If performance is not the primary system consideration, or if it

is but data memory accesses are an insignificant part of the overall processing requirements, the use of a stretch cycle may allow a more cost effective solution.

**RECOMMENDED RAM CONFIGURATION** Table 3

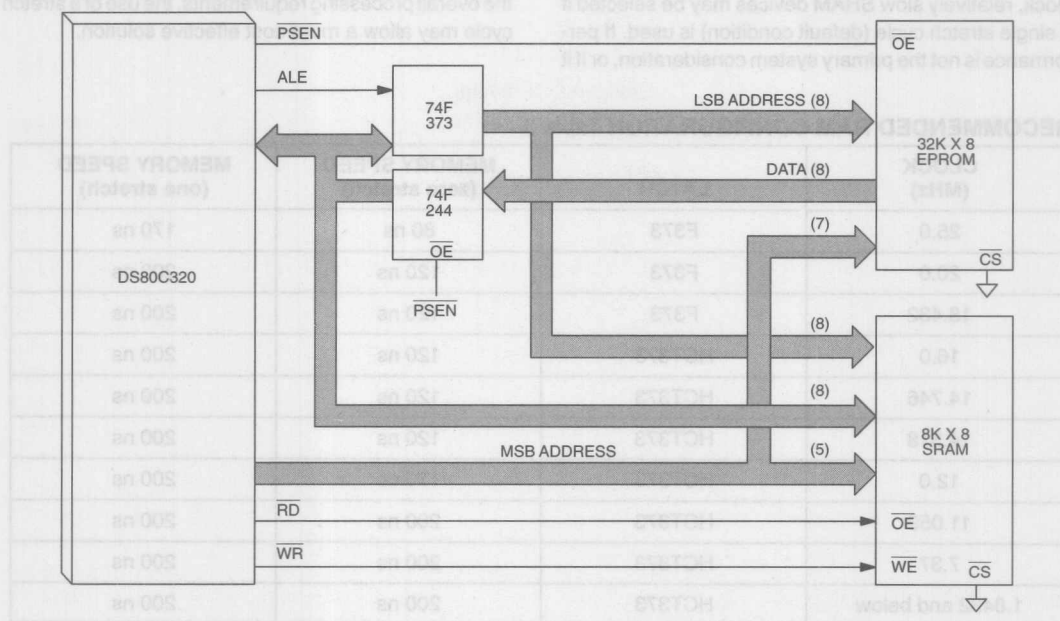
CLOCK (MHz)	LATCH	MEMORY SPEED (zero stretch)	MEMORY SPEED (one stretch)
25.0	F373	80 ns	170 ns
20.0	F373	120 ns	200 ns
18.432	F373	120 ns	200 ns
16.0	HCT373	120 ns	200 ns
14.746	HCT373	120 ns	200 ns
14.318	HCT373	120 ns	200 ns
12.0	HCT373	170 ns	200 ns
11.059	HCT373	200 ns	200 ns
7.37	HCT373	200 ns	200 ns
1.8432 and below	HCT373	200 ns	200 ns

#### ADDITIONAL CONSIDERATIONS

In developing this application note, it was noted that some EPROM devices have extremely long "turn off" times. If the EPROM selected for 25 MHz systems has an "output disable to float" time greater than 35 ns (parameter  $t_{PXIZ}=t_{CLCL}-5$ ), bus contention will occur on the processor's AD7-AD0 bus. In most situations, this simply results in higher power consumption. However, in some situations the address setup time to the memory can be affected necessitating a faster memory. The simplest solution to this problem is to use a device with the required turn-off time, but another possible solution exists. A 74F244 driver can be placed between the EPROM's output and the processor's data bus as shown in Figure 5. The 74F244's outputs turn off within a maximum of 8 ns, thereby releasing the processor's bus almost immediately and eliminating the contention.

All of the timing calculations used in this application note are based on the equations in the DS80C320 data sheet. The timing specifications given in the data sheet assume an approximately equal capacitive load on the signals specified. If the configuration of Figure 1 is used, this is achieved. If, however, any signal is connected to additional loads, then the capacitive loading including the additional devices should be evaluated. If there is a significant difference, additional margins should be used in the critical path analysis, and appropriate memory speeds selected.

For older or otherwise unconventional SRAM devices, it may be wise to confirm other important timing parameters such as data setup before write active ( $<t_{WLWH}-t_{QVWX}=75-5=70$ ). With the devices surveyed, meeting the four parameters discussed above will qualify the device for use.

**FAST EPROM TURNOFF Figure 5****EQUATION SUMMARY**

For the user who wishes to calculate the memory speed requirements using a crystal frequency not shown in the preceding tables, the following equations provide a concise summary of the information needed. The memory

devices selected must have an address access time (based on the use of an F373 or an HCT373), an  $\overline{OE}$  access time, a  $\overline{WE}$  time, and a bus release time less than or equal to the calculated values. Note again that  $t_{CLCL}$  is the period of the clock.

**EPROM Access Times**

$$\begin{aligned} \overline{OE} &= t_{PLIV} \\ &= 2.25t_{CLCL} - 20 \end{aligned} \quad \begin{aligned} \text{F373} \\ \text{address} &= t_{AVIV1} - 8 \\ &= 3t_{CLCL} - 28 \end{aligned}$$

$$\begin{aligned} \text{HCT373} \\ \text{address} &= t_{AVIV1} - 44 \\ &= 3t_{CLCL} - 64 \end{aligned} \quad \begin{aligned} \text{Bus Release} &= t_{RHDZ} \\ &= t_{CLCL} - 5 \\ &(\text{same for RAM}) \end{aligned}$$

**RAM Read Access Times**

$$\begin{aligned} \overline{OE} &= t_{RDLY} \\ &= (2.0t_{CLCL} - 20 + t_{MCS}) \end{aligned} \quad \begin{aligned} \text{F373} \\ \text{address} &= t_{AVDV1} - 8 \\ &= (3.0t_{CLCL} - 20 + t_{MCS}) - 8 \end{aligned} \quad \begin{aligned} \text{HCT373} \\ \text{address} &= t_{AVDV1} - 44 \\ &= (3.0t_{CLCL} - 20 + t_{MCS}) - 44 \end{aligned}$$

**RAM Write Pulse Time**

$$\begin{aligned} \overline{WE} &= t_{WLWH} \\ &= (2t_{CLCL} - 5 + t_{MCS}) \end{aligned}$$

# DALLAS SEMICONDUCTOR

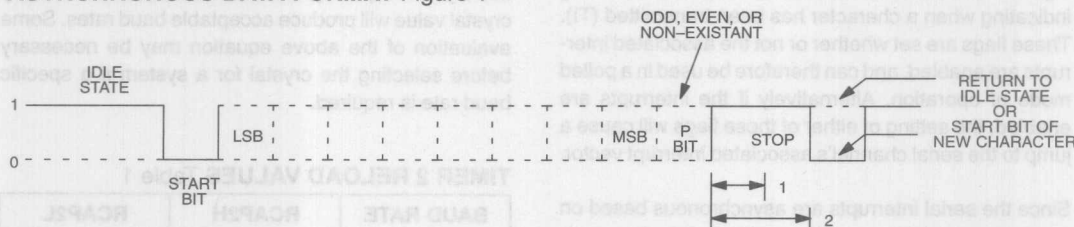
## Application Note 75 Using the High-Speed Micro's Serial Ports

### INTRODUCTION

The High-Speed Micro's serial interfaces are functionally identical to those found on other lower performance 8051 processors. They are based on a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) implementation. As the name implies, a USART converts parallel data to and from a synchronous or asynchronous serial bit stream. The parallel data side of the device interfaces with the processor's internal data bus, and the serial side interfaces with the outside world. This application note describes the setup and operation of the most commonly encountered operating modes of this interface.

Because of its universal applicability, the most frequently used configuration of the High-Speed Micro's serial channel is its 10-bit asynchronous mode. In this application note, this configuration will be described in detail. A general overview of the port's operation will be provided and a detailed software example will be presented.

### ASYNCHRONOUS DATA FORMAT Figure 1



Please note that many members of the High-Speed Microcontroller family have two functionally identical serial ports. As new members are added to the processor family, some may not have both serial ports. Refer to the individual device's data sheet for specific features. This application note will concentrate on the use of a single serial port (port 0), but in the examples, the ideas presented are equally applicable to both ports. One example program demonstrates the use of both serial ports.

Examples illustrating the use of dual serial ports and 11-bit address recognition capability will also be presented. Through these three examples, different aspects of serial port operation will be discussed.

The most frequently encountered serial communication modes are based on asynchronous data transmission. In this mode of transmission, there is no separate clock signal. The classical serial asynchronous data communications format illustrated in Figure 1 provides the necessary synchronization without a clock signal. In this format, 8 or 9 data bits are accompanied by one start bit and one or two stop bits. The 9th data bit is frequently used as a parity bit. The start and stop bits provide the necessary synchronization information. The serial bit stream's content in this mode is compatible with the popular RS-232 protocol. However, the signal levels are not. For signal level compatibility, a level translator such as the DS1228 must be used to convert TTL/CMOS levels to/from RS-232 levels.

### BAUD CLOCK SOURCES

While it is not the purpose of this document to discuss details of the High-Speed Micro's internal timers, there must be some discussion of them since they are frequently used as the source of the baud rate clock. The High-Speed Micro contains three internal timers; two of which may be used as baud rate generators. The following sections briefly describe these timers and their modes most commonly used for baud rate generation. Detailed information on the timers' setup will be illustrated by the software examples that follow.

by the processor. Once initialized, the timers run automatically, producing a baud rate clock based on a value loaded into the timers' reload registers. When in this mode, timer 1 uses SFR registers TL1 to count and TH1 to store a reload value. Software must initialize TH1 with the desired reload value, and if the first time interval is to be correct, TL1 must also be loaded with the same value. Timer 2 uses registers TL2 and TH2 for counting, and registers RCAP2L and RCAP2H for holding the reload values. Again, these registers must be initialized by software. When enabled, the timers will begin counting based on the selected clock source. The clocks possible for timer 1 are oscillator/12 or oscillator/4. For timer 2, oscillator/2 is the only possibility when in baud clock generation mode. When the count registers roll over from their maximum count to 0, they will be automatically reloaded with the value in the reload registers. The reload value remains unchanged unless modified by software. Every time the rollover occurs, a clock pulse is generated. This clock pulse is used by the serial port as a baud rate clock. By changing the reload value, a wide variety of baud rates may be achieved.

#### POLLED VS. INTERRUPT DRIVEN MODES

The High-Speed Micro's serial channels have flag bits in the SFR address space that indicate the status of the channel. For each serial channel, there is a flag indicating when a character has been received (RI) and a flag indicating when a character has been transmitted (TI). These flags are set whether or not the associated interrupts are enabled, and can therefore be used in a polled mode of operation. Alternatively if the interrupts are enabled, the setting of either of these flags will cause a jump to the serial channel's associated interrupt vector.

Since the serial interrupts are asynchronous based on serial communications with an external device, most applications will benefit from using an interrupt driven communications scheme. In this way, the processor can accomplish other tasks while it is waiting to receive an interrupt. If a polling method were used, time would be consumed continually checking the flag bits to see if one has been set. The examples of this application note illustrate both approaches. The first example demonstrates a typical interrupt driven mode of operation. The second example executes a very structured set of events, so it is well suited to polled operation.

The High-Speed Micro's mode of operation is arguably the most frequently used method of serial communication on the 8051 family. This is because this mode is compatible with the familiar RS-232 protocol. While the signal levels are different, the use of a simple level translator will allow communications with the standard serial ports of any personal computer. In fact, all of the software in this applications note was tested using a PC to interface to a DS80C320 test board.

This particular serial mode can use timer 1 to generate baud rates for serial port 1 or timers 1 or 2 for serial port 0. In this example, timer 2 is operated in auto-reload mode to generate the baud rate for serial port 0.

After establishing the timer's mode of operation, the reload value must be stored in the reload register. The contents of the reload registers may be calculated for a desired baud rate and oscillator frequency using the following equation:

$$RCAP2H, RCAP2L = 65536 - \frac{\text{Oscillator Frequency}}{32 * \text{Baud Rate}}$$

Using this equation, the reload value for any desired baud rate and oscillator frequency may be calculated. For this software example, an oscillator frequency of 11.0592 MHz is assumed. The table below shows the reload values for several common baud rates based on this crystal frequency. It should be noted that not every crystal value will produce acceptable baud rates. Some evaluation of the above equation may be necessary before selecting the crystal for a system if a specific baud rate is required.

**TIMER 2 RELOAD VALUES** Table 1

BAUD RATE	RCAP2H	RCAP2L
57600	0FFh	0FAh
9600	0FFh	0DCh
2400	0FFh	090h
1200	0FEh	0E0h

Once a reload value is determined, it must then be loaded into the timer's reload registers to establish the timer's output clock frequency.



After initializing the timer, the serial port may be set up as desired. To establish the correct operating mode for serial port 0, the SM0 and SM1 bits of the SCON register (address 098h) must be set appropriately. The following

table shows the possible settings of these bits and the resulting mode. As shown, SM0 and SM1 (SCON.1 and SCON.0) must be set to 0 and 1 respectively for 10-bit asynchronous operation.

**SERIAL MODE BITS** Table 2

SM0	SM1	MODE	FUNCTION	LENGTH	PERIOD
0	0	0	Sync	8 bits	4/12 t <sub>CLK</sub>
0	1	1	Async	10 bits	Timer 1 or 2*
1	0	2	Async	11 bits	64/32 t <sub>CLK</sub>
1	1	3	Async	11 bits	Timer 1 or 2*

\* Timer 2 is only available for baud rate generation on serial port 0.

Since the serial port will be operated in interrupt driven mode, the interrupt enables must be set appropriately. By setting the ES0 (address 0ACh) and EA (address 0AFh) bits to 1, serial port 0 will generate an interrupt when a character has been transmitted or when a character has been received.

As a final step in initialization, the timer is started by setting bit TR2 (address 0CAh). At this point, serial communications may begin. To transmit a character, the character is written to the SBUF (byte address 099h) and other tasks are performed until an interrupt is received (in this case a tight loop). In receiving a character, no action is required until an interrupt occurs. Since either a transmit or a receive can cause a jump to the same interrupt vector, the interrupt service routine (ISR) must determine which was the cause. This is done by reading the TI (bit address 099h) and RI (bit address 098h) status bits of the SCON register. If TI is set, then a "Transmit Complete" caused the interrupt. If RI is set, then a "Receive" caused the interrupt. If a transmit caused the interrupt, the TI bit may be cleared and the ISR exited. If a receive caused the interrupt, then the RI bit must be cleared and the received character must be read from SBUF.

The software listing for example 1 below illustrates the details of implementing this mode of serial operation.

### DUAL SERIAL PORT EXAMPLE

This example demonstrates the use of the two serial ports available on the DS80C320. The main purpose of the example is to illustrate how to initialize and use the second port. As discussed earlier, much of the information regarding serial port 0 applies equally to serial port

1. However, this example will help clarify any confusion about the use of this resource.

In this example, the output of port 1 (TXD1) is connected to its input (RXD1) in a "loop back" configuration. The software is written to create a closed serial loop with port 0 as input and output. A terminal or PC running a terminal emulator is connected to port 0's input (RXD0) and output (TXD0). Initially, software outputs a three line message to the terminal on port 0, and the DS80C320 waits for an input. When the terminal sends a character to the DS80C320, the RI bit is set. When the software recognizes this bit has been set, it reads the received character and transfers it to the transmit buffer of port 1. For illustrative purposes, the character is converted to upper case (if not already) before it is transferred. Since the output of port 1 is tied to its input, the transmitted character automatically ends up in the receive buffer. The software then copies this character to port 0's transmit buffer, which causes it to be transmitted out of the processor. Finally, the character arrives at the terminal as an upper case character, thereby completing the loop.

In this software example, both serial ports are set to run from a baud clock generated from Timer 1. The timer is set for auto-reload mode, and the count and reload registers are loaded with the appropriate value. Timer 1's equation for calculating reload values is different from Timer 2's, and is shown below:

$$\text{Reload} = 256 - \frac{2^{\text{SMOD}} * \text{OscillatorFreq.}}{384 * \text{BaudRate}}$$

Using the above equation, the reload value for a desired baud rate and oscillator frequency may be calculated. It can be seen that the reload value is a function of 2 raised to the power of SMOD. Since SMOD can be either 0 or 1, this term can be either 1 or 2 ( $2^0 = 1$ ,  $2^1 = 2$ ). Therefore, setting SMOD to 1 has the effect of doubling the baud rate. Again for this software example, an oscillator frequency of 11.0592 MHz is assumed. Table 3 below shows the reload values calculated for several common baud rates based on this crystal frequency.

**TIMER 1 RELOAD VALUES**

BAUD RATE	SMOD	RELOAD
57600	1	FF
19200	1	FD
9600	0	FD
2400	0	F4
1200	0	E7

As seen in the above equation, the reload value is calculated based on the SMOD baud rate doubler bit's setting. If this bit is zero for the desired baud rate, it is not necessary to clear it in the initialization software because this is its reset default condition. However for clarity, the instructions to clear this bit for both ports are included in the example. Since the SMOD bit is not "bit addressable" you must write to the entire PCON register. The example code shows instructions for clearing and setting the SMOD bit using a single logical instruction. The instruction not used in the program is commented out.

In the example, the timer mode is initialized, the count and reload registers are loaded, the two SMOD bits are cleared, and the timer interrupt is disabled. This completely configures the baud clock generation. After the two serial control registers are set for the desired mode, the timer is started, and serial communication begins.

This example handles serial communication differently than illustrated by the earlier example. This example uses a polled mode to monitor serial status. Since this example is more structured in its operation, this polled approach is appropriate. The basic functions that transfer characters, are GETCH and PUTCH. Both of these functions perform a tight loop waiting for the appropriate flag to be set. When it is, the program continues. This would normally be considered a waste of time, but in this

application and others like it, polled operation makes sense.

The software listing for example 2 illustrates the details of implementing this mode of serial operation.

### ADDRESS RECOGNITION EXAMPLE

This example demonstrates the address recognition capability of the High-Speed Micro's serial channel. In addition, it illustrates a method of baud clock generation that does not involve the use of a timer (i.e., serial mode 2).

The address recognition feature of the High-Speed Micro is frequently used for multi-processor communications. Each processor on the bus can be assigned a unique address. When configured, the processors will not recognize any serial communications unless its address is matched. Full details of this mode of operation may be found in the High-Speed Microcontroller User's Guide.

In this example, the address is set to recognize a control C character (03h). Any character received before a control C is ignored. However when a control C is received, a character string is immediately printed, and subsequent characters received on RXD0 are echoed out TXD0.

As mentioned, the illustrated mode of generating a baud rate does not involve a timer (i.e., serial mode 2). The baud rate is selectable as shown in the following equation:

$$\text{baudrate} = \frac{2^{\text{SMOD}} * \text{OscillatorFreq.}}{64}$$

As can be seen in the equation, the selection of crystal frequencies is much more restricted for this serial mode than others if a particular baud rate is desired. In fact, there are relatively few crystal frequencies available that will produce a standard baud rate. In this example, an oscillator frequency of 7.372 MHz is assumed which will result in a rate of 115,200 baud with SMOD cleared to 0. It is quite common to see an oscillator of 12.0 MHz that will produce a rate of 187,500 baud with SMOD cleared to 0.

The software listing for example 3 illustrates the details of implementing this mode of serial operation.

**CODE EXAMPLE # 1 : 10-BIT ASYNCHRONOUS MODE**

```

; * * * * *
; * * * * *      Program ASYNC10B.ASM * * * * *
; * * * * *
;
;   This program sets up the DS80C320's Port 0 serial port to operate
;   in 10-bit asynchronous format. Timer 2 is used as the baud rate
;   generator by setting it in auto-reload mode. A crystal value of
;   11.0529 MHz is assumed for the baud rate reload values.
;
;
;
; * * * * *
; * * * * *
;
;           Interrupt Vectors
;
;           ORG      0000h          ; RESET VECTOR
;           AJMP     START          ; Jump to start of program
;
;           ORG      0023h          ; SERIAL PORT 0 VECTOR
;           AJMP     SERINT         ; Jump to serial interrupt routine
;
; * * * * *
;
;           Initialization
;
;           ORG      0100h
;
; START:    MOV      IE, #0          ; Disable ALL interrupts
;
;           Set up timer 2
;
;           MOV      T2CON, #030h   ; Timer 2 : auto-reload mode for
;                                   ; both receive & transmit baud clocks
;           MOV      RCAP2L, #T2RL96L ; Establish 16-bit reload value for 9600
;           MOV      RCAP2H, #T2RL96H
;           MOV      TL2, #T2RL96L ; Make first timeout correct
;           MOV      TH2, #T2RL96H
;
; * * * * *
;
;           Set up Serial Port 0
;
;           MOV      SCON, #050h    ; Mode 1, set receive enable
;           SETB     ES0             ; Enable Serial Port interrupt
;
;           SETB     EA              ; Set Global Interrupt enable
;           SETB     TR2             ; Start Timer
;
;
; * * * * *
; * * * * *      Test Code to exercise the serial port * * * * *
;
;           MOV      SBUF, #'D'      ; Put character 'D' in buffer (send it)
;           CALL     WAIT            ; Wait until flag cleared from interrupt
;                                   ; service routine.
;
;           MOV      SBUF, #'S'
;           CALL     WAIT

```

```

MOV     SBUF, #'8'
CALL    WAIT
MOV     SBUF, #'0'
CALL    WAIT
MOV     SBUF, #'C'
CALL    WAIT
MOV     SBUF, #'3'
CALL    WAIT
MOV     SBUF, #'2'
CALL    WAIT
MOV     SBUF, #'0'
CALL    WAIT
MOV     SBUF, #00Dh
CALL    WAIT
MOV     SBUF, #00Ah
CALL    WAIT
;
SJMP    $

;
; * * * * *
; * * * * * Subroutine to wait for a serial interrupt * *
WAIT:    MOV     R1, #0FFh
HERE:    CJNE    R1, #0000h, HERE
RET

;
; * * * * *
; * * * * * Serial Interrupt Service Routine * * * * *
;
SERINT:   CLR     ES0             ; Disable serial interrupts
;
JNB      TI, SER_A             ; If TI=0, must be receive complete interrupt
CLR      TI                   ; Else must be transmit complete interrupt
SJMP     SER_X

SER_A:    CLR     RI             ; Clear the receive bit
MOV      A, SBUF              ; Get the character and
MOV      SBUF, A              ; echo it

SER_X:    MOV     R1, #00h       ; Indicate interrupt has been serviced
; (i.e. exit wait)
SETB     ES0                  ; Re-enable serial interrupts
RETI

;
; * * * * *
;
END
; Program ASYNC10B.ASM

```

## CODE EXAMPLE #2 : DUAL SERIAL PORT

```

; *****
; ***** Program Tst2Ser *****
; *****
;
; This program demonstrates the simultaneous use of both of the DS80C320's
; serial ports. Both serial channels run off of timer 1 which is set for
; 9600 Kbaud with SMOD = 0, and an 11.059 MHz crystal. The program
; assumes that the transmit and receive pins of port 1 are connected
; together (i.e., TXD1 tied to RXD1). When the program starts, the
; DS80C320 sends a three line message out on TXD0. It then waits for
; a character to come in on RXD0 from an external terminal. This
; received character is converted to upper case if it is not already.
; The upper case character is then transferred to the transmit pin of
; port 1, TXD1, and the processor then waits for a character to be
; received on port 1's receive pin, RXD1. The received character is
; then transferred to port 0's transmit pin, TXD0, which completes the
; loop with the terminal.
;
; *****
; *****
;
; Initialization
;
; ORG 0100h
START:
; MOV IE, #0 ; Disable all interrupts
;
; Set up Timer 1
;
; MOV TMOD, #020h ; Timer 1: Mode 2 (8-bit reload)
; MOV TH1, #REL96 ; Reload value for 9600, SMOD = 0
; MOV TL1, #REL96 ; Make first time-out correct
;
; ANL (PCON), #07Fh ; Clear SMOD bit for port 0
; ANL (WCON), #07Fh ; Clear SMOD bit for port 1
;
; CLR ET1 ; Disable timer 1 interrupts
;
; *****
;
; Set up serial ports
;
; MOV SCON, #050h ; Timer 1 : Mode 1 (vari baud)
; ; and set receive enable
; MOV SCON1, #050h ; Timer 2 : Mode 1 (vari baud)
; ; and set receive enable
; SETB TR1 ; Start timer 1
;
; *****
;
; Configuration Demonstration
;
; ; Output 3 Message Lines
; MOV DPTR, #MSG1 ; Point to first message
; CALL PUTS ; and call routine to output
; MOV DPTR, #MSG2 ; Point to second message

```



```

CALL PUTS      ; and call routine to output
MOV DPTR, #MSG3 ; Point to third message
CALL PUTS      ; and call routine to output
;
GETCH:
JNB RI, GETCH  ; Wait here for received character
CLR RI        ; Get ready for next character
MOV A, SBUF    ; Put new character in Acc
;
; If lower case character, convert to upper case
CJNE A, #'a', GETCH_A ; If character
GETCH_A: JC GETCH_CONT ; is between 'a'
CJNE A, #'z'+1, GETCH_B ; and 'z' +1
GETCH_B: JNC GETCH_CONT ; it is lower case
CLR C        ; so subtract 020h
SUBB A, #020h ; from it
;
GETCH_CONT:
CALL PUTCH1    ; Output character to port 1
;
GETCH1:
JNB RI1, GETCH1 ; Wait here for received character
CLR RI1        ; Get ready for next character
MOV A, SBUF1    ; Put new character in Acc
GETCH1_CONT:
CALL PUTCH     ; Output character to port 0
;
JMP GETCH      ; Repeat cycle
;
;
; *****
; ***** Subroutines *****
; *****
;
; ; Output a character on port 0
PUTCH:
MOV SBUF, A    ; Copy Acc to SBUF 0
JNB TI, $      ; Wait here until TI bit set
CLR TI        ; Clear TI for next character
RET           ; Return
;
; *****
;
; ; Output a string on port 0
PUTS:
CLR A          ; Make address offset zero
MOVC A, @A+DPTR ; Get next char to output
JZ PUTS_A     ; If zero, end of string
CALL PUTCH    ; Output character pointed to
INC DPTR      ; Point to next character
JMP PUTS      ; Repeat procedure
PUTS_A:
RET           ; Exit from routine
;
; *****
;
; ; Output a character on port 1
PUTCH1:

```

```

MOV     SBUF1, A      ; Copy Acc to SBUF 1
JNB     TI1, $        ; Wait here until TI bit set
CLR     TI1          ; Get ready for next character
RET     ; Return

; *****
;
; ; Output a string on port 1
PUTS1:
CLR     A            ; Make address offset zero
MOVC    A, @A+DPTR    ; Get next char to output
JZ      PUTS1_A       ; If zero, end of string
CALL    PUTCH1        ; Output character pointed to
INC     DPTR          ; Point to next character
JMP     PUTS1         ; Repeat procedure

PUTS1_A:
RET     ; Exit from routine

; *****
; *****
;
msg1:   db    'Dallas Semiconductor DS80C320', 0dh, 0ah, 0
msg2:   db    'Serial Port Tests.', 0dh, 0ah, 0
msg3:   db    'Characters typed in after this are converted to uppercase.'
        db    0dh, 0ah, 0

; *****
; *****
;
END     ; Program Tst2Ser

```

```

; * * * * * Program ADRECMD.ASM * * * * *
;
; This program sets up the DS80C320's Port 0 serial channel to operate
; in Multi-Processor Communications mode. In this mode, serial
; characters received that do not match a masked address are ignored
; (no interrupt generated). The address and mask are initially set to
; recognize a <cntl>C character (03h). When a <cntl>C is received, a message
; is sent out, and characters input are "echoed" out unchanged.
;
; * * * * *
; * * * * *
; Interrupt Vectors
;
; ORG 0000h ; RESET VECTOR
; AJMP START ; Jump to start of program
;
; ORG 0023h ; SERIAL PORT 0 VECTOR
; AJMP SERINT ; Jump to serial interrupt routine
;
; * * * * *
; Initialization
;
; ORG 0100h
START:
; MOV IE, #0 ; Disable ALL interrupts
; MOV P1, #0 ; Clear ports P1
;
; Clear SMOD for divide by 64
; ANL (PCON), #07Fh ; Clear SMOD bit
;
; * * * * *
;
; Set up Serial Port 0
;
; MOV SCON, #0B8h ; Mode 2, set receive enable,
; ; set multi-cpu communication, and set
; ; TB8 to be a 1 (consistant with input)
; SETB ES0 ; Enable Serial Port interrupt
;
; MOV SADDR0, #03h ; Set address register to <cntl>C (03h)
; MOV SADEN0, #0FFh ; Set mask to all 1s (no mask)
;
; SETB EA ; Global Interrupt enable
;
; * * * * *
; * * * * * Code to exercise the serial port * * * * *
;
; LOOP:
; MOV A, SADEN0 ; If <cntl>C has not been received
; CJNE A, #00, LOOP ; continue waiting here.
; CALL WAIT ; <cntl>C will be echoed, wait till
; ; transmit complete.
;
; ; Output a message
; MOV SBUF, #'D' ; Put character 'D' in buffer (send it)
; CALL WAIT ; Wait until flag cleared from interrupt
; ; service routine.

```

```

MOV    SBUF, #'S'      ; Repeat for 'S'
CALL   WAIT
MOV    SBUF, #'8'      ; Repeat for '8'
CALL   WAIT
MOV    SBUF, #'0'      ; Repeat for '0'
CALL   WAIT
MOV    SBUF, #'C'      ; Repeat for 'C'
CALL   WAIT
MOV    SBUF, #'3'      ; Repeat for '3'
CALL   WAIT
MOV    SBUF, #'2'      ; Repeat for '2'
CALL   WAIT
MOV    SBUF, #'0'      ; Repeat for '0'
CALL   WAIT
MOV    SBUF, #00Dh     ; Repeat for <cr>
CALL   WAIT
MOV    SBUF, #00Ah     ; Repeat for <lf>
CALL   WAIT
;
SJMP   $               ; Continuous loop. Only serial
                        ; interrupts exit loop. Characters are
                        ; echoed.
;
; *****
; ***** Subroutine to wait for a serial interrupt *****
WAIT:   MOV    R1, #0FFh ; Serial ISR clears R1 when complete
HERE:   CJNE   R1, #0000h, HERE
        RET
;
; *****
; ***** Serial Interrupt Service Routine *****
SERINT: CLR    ES0      ; Disable serial interrupts
;
        JNB    TI, SER_A ; If TI=0, must be receive complete interrupt
        CLR    TI        ; Else must be transmit complete interrupt
        SJMP   SER_X
;
SER_A:   CLR    RI        ; Clear the receive bit
        MOV    A, SBUF    ; Get the character
        MOV    SADEN0, #0 ; Clear address mask register
        MOV    P1, A      ; Display character on port 1
        MOV    SBUF, A    ; and echo it
;
SER_X:   MOV    R1, #00h   ; Indicate interrupt has been serviced
                        ; (i.e. exit wait)
        SETB   ES0        ; Re-enable serial interrupts
        RETI             ; Return from interrupt
;
; *****
; *****
END      ; Program ADRECMD

```

**APPENDIX A : CONSTANT DEFINITIONS**

```

; * * * * *
; * * * * * DEFINITIONS * * * * *
;
;   Define Register Addresses
PCON      EQU    087h      ; Contains SMOD baud rate doubler bit
TCON      EQU    088h      ; Control for timer 1 and 0
TMOD      EQU    089h      ; Mode register for timers 1 and 0
TL1       EQU    08Bh      ; Timer 1 low (count) byte
TH1       EQU    08Dh      ; Timer 1 high (reload) byte
CKCON     EQU    08Eh      ; Clock Control register
P1        EQU    090h      ;
SCON      EQU    098h      ; Control for serial port 0
SBUF      EQU    099h      ; Data buffer for serial port 0
IE        EQU    0A8h      ; Interrupt enables
SADDR0    EQU    0A9h      ;
SADEN0    EQU    0B9H      ;
SCON1     EQU    0C0h      ; Control for serial port 1
SBUF1     EQU    0C1h      ; Data buffer for serial port 1
T2CON     EQU    0C8h      ; Timer 2 control register
RCAP2L    EQU    0CAh      ; Timer 2 low byte of reload register
RCAP2H    EQU    0CBh      ; Timer 2 high byte of reload register
TL2       EQU    0CCh      ; Timer 2 low byte of count register
TH2       EQU    0CDh      ; Timer 2 high byte of count register
WDCON     EQU    0D8h      ; Watchdog Control register
;
;   Define bit addresses
TR1       EQU    08Eh      ; Timer 1 enable bit
RI        EQU    098h      ; Serial port 0 receive interrupt flag
TI        EQU    099h      ; Serial port 0 transmit interrupt flag
REN       EQU    09Ch      ; Serial port 0 receive enable bit
SM2       EQU    09Dh      ; Serial port 0 mode bit 2
SM1       EQU    09Eh      ; Serial port 0 mode bit 1
SM0FE     EQU    09Fh      ; Serial port 0 mode bit 0
;
RI1       EQU    0C0h      ; Serial port 1 receive interrupt flag
TI1       EQU    0C1h      ; Serial port 1 transmit interrupt flag
REN1      EQU    0C4h      ; Serial port 1 receive enable bit
SM21      EQU    0C5h      ; Serial port 1 mode bit 2
SM11      EQU    0C6h      ; Serial port 1 mode bit 1
SM0FE1    EQU    0C7h      ; Serial port 1 mode bit 0
;
TR2       EQU    0CAh      ; Timer 2 enable bit
;
ET1       EQU    0ABh      ; Timer 1 interrupt enable
ES0       EQU    0ACh      ; Serial port 0 interrupt enable
ES1       EQU    0AEh      ; Serial port 1 interrupt enable
EA        EQU    0AFh      ; Global interrupt enable
;
;   Define Constants
;   Timer 1 Reload Values
REL576    EQU    0FFh      ; Reload value for 57600 baud, SMOD = 1
REL96     EQU    0FDh      ; Reload value for 9600 baud, SMOD = 0
REL24     EQU    0F4h      ; Reload value for 2400 baud, SMOD = 0
REL12     EQU    0E7h      ; Reload value for 2400 baud, SMOD = 0
;   Timer 2 Reload Values
T2RL57H   EQU    0FFh      ; 16-bit Reload value for 57600 baud
T2RL57L   EQU    0FAh
T2RL96H   EQU    0FFh      ; 16-bit Reload value for 9600 baud

```



```
T2RL96L EQU 0DCh
T2RL24H EQU 0FFh ; 16-bit Reload value for 2400 baud
T2RL24L EQU 070h
T2RL12H EQU 0FEh ; 16-bit Reload value for 1200 baud
T2RL12L EQU 0E0h
;
; * * * * *
```

APPENDIX B: MAXIMUM BAUD RATES

It is frequently asked, what the maximum possible baud rate is for a particular serial mode of the DS80C320. Assuming a 25MHz crystal, the following table shows the maximum possible baud rates for the four primary serial modes.

MODE	MAXIMUM BAUD, 25 MHz	MAXIMUM BAUD, 33 MHz
0	6,250,000	8,250,000
1	390,625	515,625
2	781,250	1,031,250
3	390,625	515,625

Because the power management features in conjunction with many of the peripheral functions, especially the interrupt and serial functions, the user is urged to become familiar with the overall operation of the processor before beginning this section.

The DS87C50 incorporates a number of new features specifically designed for power management. They were designed to reduce power consumption without sacrificing throughput or responsiveness to external events. These features are listed below:

DYNAMIC CLOCK SPEED CONTROL

The High-Speed Microcontroller support four clock management modes: Stop (PMM1 (Power Management Mode 1), PMM2 (Power Management Mode 2), and Idle. The DS87C50 can dynamically switch between these modes, allowing the user to optimize the speed of the device while minimizing power consumption. The Stop mode has been improved over standard 8086 capabilities, and now supports resume from external interrupts as well as reset sources.

SWITCHBACK

The DS87C50 incorporates an automatic "switchback" feature to allow a device operating in a Power Management Mode (PMM) to switch into "high gear" upon receipt of an external interrupt or serial port transmission. This enables a device in power-saving mode to respond quickly to external events and/or operate at

features designed into the DS87C50 High-Speed Microcontroller further reduce power consumption.

Dallas Semiconductor microcontrollers are manufactured with a Complementary Metal Oxide Semiconductor (CMOS) process which makes them inherently low power devices. Unlike other technologies, CMOS devices have an infinitesimal quiescent current and only draw significant currents when switching logic states. This means that without further intervention, the user has already designed a low power system. The power management features of the DS87C50 family of microprocessors allow the system designer to effect an even greater reduction in power. The maximum current consumption in both operating and halted states of the microcontroller is shown below.

Maximum Current 25 MHz
Stopped Operating Speed (PMM2) Osc. Halted
DS87C520 1.2 mA
DS87C50 1.4 mA

A number of factors can affect power consumption that are generally beyond the ability of the device to control during operation. The single largest factor in power consumption of a microcontroller is clock frequency. The power consumed by a microprocessor is directly proportional to its operating speed, so it follows that a device operating at the lowest possible frequency will produce the maximum power savings. The speed also depends on the system requirements; most notably interrupt service time. Temperature can also affect power consumption. Semiconductor devices draw

# DALLAS SEMICONDUCTOR

## Application Note 78 Using Power Management with the DS87C5x0

### OVERVIEW

Power management is critical in battery-powered applications. Differences of microamperes can translate into months or years of operating life, which can make or break a product in the marketplace. The high level of integration of Dallas Semiconductor microcontrollers make them ideal for portable or battery-operated applications which demand low power consumption. By combining the processor and peripherals onto a single die, redundant hardware is eliminated, and power savings are achieved. In addition, power management features designed into the DS87C5x0 High-Speed Microcontrollers further reduce power consumption.

Dallas Semiconductor microcontrollers are manufactured with a Complementary Metal Oxide Semiconductor (CMOS) process which makes them inherently low power devices. Unlike other technologies, CMOS devices have an infinitesimal quiescent current and only draw significant currents when switching logic states. This means that without further intervention, the user has already designed a low power system. The power management features of the DS87C5x0 family of microprocessors allow the system designer to effect an even greater reduction in power. The maximum current consumption in both operating and halted states of the microcontroller is shown below.

Maximum Current, 25 MHz

	<u>Slowest Operating Speed (PMM2) Osc. Halted</u>
DS87C520	1.2 mA
	1 $\mu$ A

A number of factors can affect power consumption that are generally beyond the ability of the device to control during operation. The single largest factor in power consumption of a microcontroller is clock frequency. The power consumed by a microprocessor is directly proportional to its operating speed, so it follows that a device operating at the lowest possible frequency will produce the maximum power savings. The speed chosen depends on the system requirements, most notably interrupt service time. Temperature can also affect power consumption. Semiconductor devices draw

greater power at lower temperatures. If the system under development is being designed for cold temperatures, the designer should expect higher than typical power consumption values. System design also has a direct bearing on power consumption, and driving large external loads will increase power consumption.

This application note covers the DS87C520 and DS87C530 High-Speed Microcontrollers. Their power management features are explained, and techniques are presented for minimizing power consumption. Because the power management feature operates in conjunction with many of the peripheral functions, especially the interrupt and serial functions, the user is urged to become familiar with the overall operation of the processor before beginning this section.

The DS87C5x0 incorporates a number of new features specifically designed for power management. They were designed to reduce power consumption without sacrificing throughput or responsiveness to external events. These features are listed below:

### DYNAMIC CLOCK SPEED CONTROL

The High-Speed Microcontrollers support four clock management modes: Stop, PMM1 (Power Management Mode 1), PMM2 (Power Management Mode 2), and Idle. The DS87C5x0 can dynamically switch between these modes, allowing the user to optimize the speed of the device while minimizing power consumption. The Stop mode has been improved over standard 8051 capabilities, and now supports resume from external interrupts as well as reset sources.

### SWITCHBACK

The DS87C5x0 incorporates an automatic "switchback" feature to allow a device operating in a Power Management Mode (PMM) to switch into "high gear" upon receipt of an external interrupt or serial port transmission. This enables a device in power-saving mode respond quickly to external events and/or operate its

serial ports. Traditional 8051-based devices without the switchback feature lose the ability to service interrupts quickly without running the device at high speed, and higher power consumption, constantly.

### SELECTABLE CLOCK SOURCE

The crystal oscillator is a large consumer of power on any microcontroller, especially during low power operation. The DS87C5x0 ring oscillator, used for quick starts from Stop mode, can also be used to provide an approximately 3 to 4 MHz clock source during normal operation. Although a crystal oscillator is still required at power-up, once the crystal has stabilized, device operation can be switched to the ring oscillator, realizing a power savings of as much as 25 mA.

### BAND-GAP REFERENCE DISABLING

The DS87C5x0 gives the user the option of disabling the band-gap reference, which is used to detect a power failure while in Stop mode. Stop mode current can be reduced from 80  $\mu$ A to 1  $\mu$ A by using this feature.

### ENHANCED STATUS REPORTING

Although the ability to dynamically switch the internal clock speed is a benefit, if performed at the wrong time it can seriously interfere with the operation of timing-dependent functions. The Status register (STATUS;C5h), new to the DS87C5x0 devices, contains information about the status of both serial ports, the crystal oscillator, and high priority, low priority, and power fail interrupts. The software can delay or cancel a planned speed change based on the information in this register.

### CLOCK SPEED CONTROL

#### Description

The operating frequency of a microcontroller is the single biggest factor in determining power consumption. The DS87C5x0 family of microcontrollers supports four clock speed management modes which conserve power by slowing or stopping the internal clock. These modes allow the system designer to maximize power savings with a minimum impact on performance.

#### PMM1

Power Management Mode 1 (PMM1) allows the user to run the DS87C5x0 at a reduced speed to save power. Setting the clock divider rate bits (PMR.7–6) will force the part from its default 4 clocks per machine cycle

(divide by 4) to 64 clocks per machine cycle (divide by 64). The external crystal continues to operate at full speed. All peripherals and instructions will operate at this reduced speed. The DS87C5x0 can resume divide by 4 operation by setting the appropriate clock divider rate bits or by utilizing the switchback feature.

#### PMM2

Power Management Mode 2 (PMM2) allows the user to run the DS87C5x0 at an even slower speed to improve power savings. Setting the clock divider rate bits (PMR.7–6) will force the part from its default 4 clocks per machine cycle (divide by 4) to 1024 clocks per machine cycle (divide by 1024). The external crystal continues to operate at full speed. All peripherals and instructions will operate at this reduced speed. The DS87C5x0 can resume full-speed (divide by 4) operation by setting the appropriate clock divider rate bits or by utilizing the switchback feature. This mode permits an even greater power savings over PMM1.

### STOP MODE

The Stop mode is the lowest power state available to the DS87C5x0. It is initiated by setting the Stop bit (PCON.1). While in this mode the crystal oscillator is stopped, and all internal clocking, including the Watchdog Timer, is halted. The real time clock on the DS87C530 is unaffected by Stop mode. The Stop mode is exited by an external interrupt, real-time clock interrupt, an external reset via the RST pin, or a power-on reset. Each interrupt will cause the device to vector to the corresponding interrupt routine to resume execution.

The DS87C5x0 incorporates a ring oscillator to allow for a fast resumption from Stop mode. This provides an instantaneously available 4 MHz clock source for the device to start operation. It can function until the crystal has stabilized, or can continue to be used as the clock source. The ring oscillator does not exhibit as much stability as an external clock, and the device should not perform timing measurements requiring high accuracy or serial port data transfers while operating from the ring oscillator. For more information concerning the use of the ring oscillator, please consult the Clock Source Control section of this document.

## IDLE MODE

The Idle mode halts operation of the DS87C5x0 processor core but leaves internal clocks, serial ports, and timers running. This mode is invoked by setting the IDL bit (PCON.0), and can be exited by an interrupt or external reset via the RST pin. Use of this mode is not recommended on new designs, as lower power operation can be achieved by placing the part in PMM2 and executing NOPs. Its inclusion in the DS87C5x0 provides backward software compatibility.

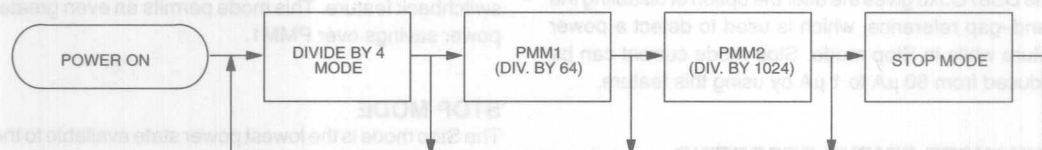
## POWER MANAGEMENT MODES

The greatest power savings come from utilizing the power management modes associated with the DS87C5x0. Unlike other techniques, Power Manage-

ment Modes 1 and 2 (PMM1 and PMM2) allow the user to reduce power consumption without sacrificing performance. Although the power management features are an important part of a power efficient design, a thorough understanding of the microprocessor will allow the system designer to achieve maximum power savings.

The clock speed management modes are designed to be part of a progressive level of power reduction, based on external activity and performance needs. PMM1 and PMM2 provide the lowest level of power consumption while still permitting full computational and peripheral operation. Figure 1 demonstrates the progression of clock management modes. As explained later, transitions between PMM1 and PMM2 must be made through divide by 4 mode.

PROGRESSION OF CLOCK SPEED MODES Figure 1



The DS87C5x0 incorporates a ring oscillator to allow for a fast resumption from Stop mode. This provides an instantaneous available 4 MHz clock source for the device to that operation. It can function until the clock has stabilized, or can continue to be used as the clock source. The ring oscillator does not exhibit as much jitter as an external clock, and the device should not perform timing measurements requiring high accuracy or serial port data transfers while operating from the ring oscillator. For more information concerning the use of the ring oscillator, please consult the Clock Source Control section of this document.

Although the ability to dynamically switch the internal clock speed is a benefit, it is performed at the wrong time if can seriously interfere with the operation of timing-dependent functions. The Status register (STATUS CSR), new to the DS87C5x0 devices, contains information about the status of both serial ports, the crystal oscillator, and high priority low priority, and power fail interrupt. The software can delay or cancel a planned speed change based on the information in this register.

## CLOCK SPEED CONTROL

### Description

The operating frequency of a microcontroller is the single biggest factor in determining power consumption. The DS87C5x0 family of microcontrollers supports four clock speed management modes which conserve power by slowing or stopping the internal clock. These modes allow the system designer to maximize power savings with a minimum impact on performance.

### Power Management Modes 1 (PMM1)

Power Management Mode 1 (PMM1) allows the user to run the DS87C5x0 at a reduced speed to save power. Setting the clock divider rate (PMM1-3) will force the part from its default 4 clock per machine cycle

## ENTERING AND EXITING POWER MANAGEMENT MODES

Software invokes the desired power management mode using bits in the Power Management Register (PMR). Stop mode is invoked by setting the STOP bit (PCON.1). The device speed is selected by the clock divider rate bits CD1, CD0 (PMR.7–6), shown below.

### CLOCK DIVISOR RATE BIT SETTINGS

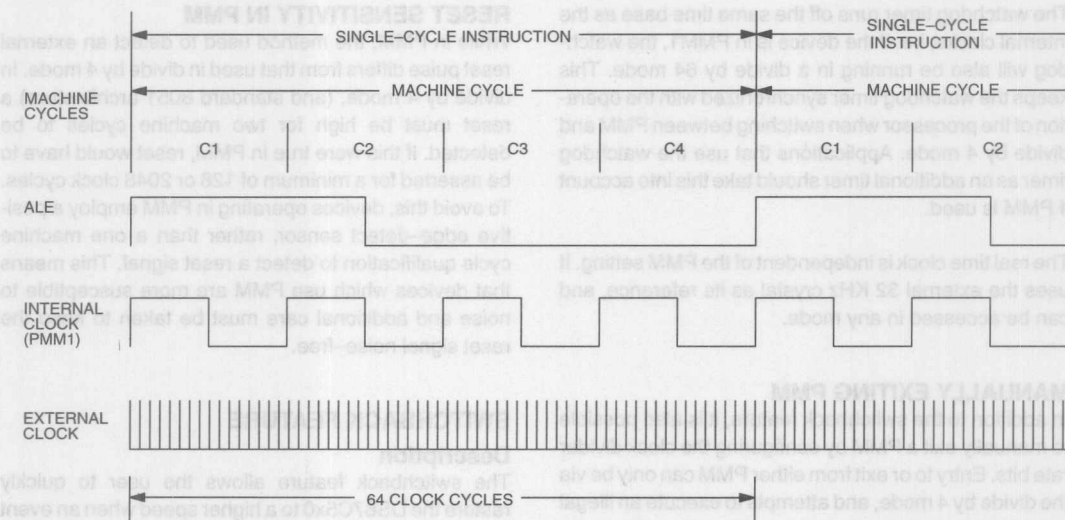
CD1	CD0	MACHINE CYCLE RATE
0	0	Reserved
0	1	4 clocks (default)
1	0	64 clocks (PMM1)
1	1	1024 clocks (PMM2)

PMM1 and PMM2 can be exited by configuring the clock divider rate bits CD1, CD0, or by the switchback function. Entry to or exit from either PMM can only be via the divide by 4 mode. For example, to go from PMM1 (divide

by 64) to PMM2 (divide by 1024) mode, it is necessary to first switch from PMM1 to divide by 4 mode, and then from divide by 4 to PMM2. Attempts to execute an illegal speed change will be ignored and the bits will remain unchanged. It is the responsibility of the software to test for serial port activity using the Status register (STATUS;C5h) before attempting to change speed. Changing speed during an asynchronous serial port operation will corrupt the serial transmission.

When PMM is invoked, the external crystal will continue to operate at full speed, and the DS87C5x0 will still execute four internal states per machine cycle. In PMM the device performs an internal divide of the external clock, by 16 for PMM1 ( $16 \times 4 = 64$ ) or 256 for PMM2 ( $256 \times 4 = 1024$ ) to achieve the desired frequency, as opposed to actually performing 64 or 256 internal states per instruction. For example, operations that occur during C2 will still do so. Most applications will not find it necessary to attend to this much detail, but the information is provided for calculating critical timings.

### INTERNAL TIMING RELATIONSHIPS IN PMM1 Figure 2





cedures. Timed Access operates in relation to internal machine cycles, not an absolute time reference.

## TIMERS AND PMM

Timers 0, 1, and 2 will default on power-up to a 12 external clocks per timer tick to remain compatible with the original 8051/8032 specifications. The timers can be

divide by 4 mode by setting the relevant bits in the Clock Control Register (CKCON;8Eh). During PMM timers 0, 1, and 2 operate at correspondingly reduced clock rates, because the timers derive their time base from the internal clock. This will also affect the operation of the serial ports in PMM as the timers are used to generate baud rates. Table 2 shows the effect of the clock divider rate on timer operation.

**EFFECT OF CLOCK MODES ON TIMER OPERATION** Table 2

CD1	CD0	OSC. CYCLES PER MACHINE CYCLE	OSC. CYCLES PER TIMER 0/1/2 CLOCK		OSC. CYCLES PER TIMER 2 CLOCK, BAUD RATE GEN.		OSC. CYCLES PER SERIAL PORT CLOCK MODE 0		OSC. CYCLES PER SERIAL PORT CLOCK MODE 2	
			TxM=1	TxM=0	T2M=1	T2M=0	SM2=0	SM2=1	SMOD=0	SMOD=1
0	0	Reserved								
0	1	4	12	4	2	2	12	4	64	32
1	0	64 (PMM1)	192	64	32	32	192	64	1024	512
1	1	1024 (PMM2)	3072	1024	512	512	3072	1024	16,384	8192

The watchdog timer runs off the same time base as the internal clocks; i.e. if the device is in PMM1, the watchdog will also be running in a divide by 64 mode. This keeps the watchdog timer synchronized with the operation of the processor when switching between PMM and divide by 4 mode. Applications that use the watchdog timer as an additional timer should take this into account if PMM is used.

The real time clock is independent of the PMM setting. It uses the external 32 KHz crystal as its reference, and can be accessed in any mode.

## MANUALLY EXITING PMM

In addition to the switchback feature, it is also possible to manually exit a PMM by configuring the clock divider rate bits. Entry to or exit from either PMM can only be via the divide by 4 mode, and attempts to execute an illegal speed change will be ignored and the bits will remain unchanged. If a timing-dependent operation may be in progress, the Status register (STATUS;C5h) should be interrogated to determine if it has been completed before switching out of or into PMM.

## RESET SENSITIVITY IN PMM

While in PMM, the method used to detect an external reset pulse differs from that used in divide by 4 mode. In divide by 4 mode, (and standard 8051 architecture) a reset must be high for two machine cycles to be detected. If this were true in PMM, reset would have to be asserted for a minimum of 128 or 2048 clock cycles. To avoid this, devices operating in PMM employ a positive edge-detect sensor, rather than a one machine cycle qualification to detect a reset signal. This means that devices which use PMM are more susceptible to noise and additional care must be taken to keep the reset signal noise-free.

## SWITCHBACK FEATURE

### Description

The switchback feature allows the user to quickly restore the DS87C5x0 to a higher speed when an event occurs. When enabled, a qualified event causes the device to automatically switch from divide by 64 (PMM1) or divide by 1024 (PMM2) to divide by 4 operation without software intervention. This allows the device to

respond to high priority or interrupt driven events with a minimum delay. The following sources can trigger a switchback:

external interrupt 0/1/2/3/4/5,  
serial start bit detected, Serial Port 0/1,  
transmit buffer loaded, Serial Port 0/1,  
watchdog timer reset,  
power-on reset,  
external reset.

Because of the intimate relationship between PMM, switchback, external interrupts, timer/counters, and the serial ports, it is highly recommended that the system designer become familiar with these features before attempting to use the switchback feature.

### Status Register

A Status register (STATUS;C5h) has been added to the DS87C5x0 to aid the software in determining whether a speed change is appropriate. The Status register provides information on the status of both serial ports, and high priority, low priority, and power fail interrupts, allowing the device to determine whether or not the device should be switched into PMM.

The benefits of the Status register become apparent when using the switchback feature to exit or enter PMM. A device executing an interrupt service routine in PMM will not execute a switchback in response to an interrupt of equal or lower priority. The Status register can be used to test for an interrupt service routine in progress, and can hold off entering PMM until finished, or take another course of action.

### Enabling/Initiating switchback

Automatic switchback is enabled by setting the SWB bit (PMR.5). When a qualified switchback event occurs, the device will exit either PMM and return to the default operating mode of 4 clocks per machine cycle. Clearing the SWB bit will disable the ability of external interrupts and serial ports to cause future switchbacks, but will not affect the current speed of the DS87C5x0. Five conditions must be met for an external interrupt to cause a switchback:

1. The device must currently be in PMM1 or PMM2.
2. The SWB bit (PMR.5) must be set.
3. Global Interrupts must be enabled by setting the EA bit (IE.7).

4. The specific interrupt must be enabled.
5. The specific interrupt occurs and is acknowledged.

Switchbacks via the serial port are slightly different. In general, switchbacks are caused by interrupts. In the case of the serial ports, this introduces a problem as they generate interrupts only upon receipt or transmission of a complete word. For the serial port to properly receive or transmit a word at standard baud rates, it must be operating at full speed. If the DS87C5x0 is operating in PMM, it would never complete a reception to initiate an interrupt, or the corresponding switchback.

The DS87C5x0 solves this problem by initiating a switchback, if enabled, upon the receipt of a falling edge on the RX pin, not the receiver interrupt. This switches the device back to full speed on the next internal machine cycle, in time to capture the start bit, and the rest of the transmission. Note that the ability of the serial port to initiate a switchback is not dependent on the Enable Serial Port Interrupt bits (IE.4 or IE.6), only the specific Receiver Enable bit (SCON0.4 or SCON1.4). Four conditions must be met for a serial port reception to cause a switchback:

1. The device must currently be in PMM1 or PMM2.
2. The SWB bit (PMR.5) must be set.
3. The specific serial port must be enabled by setting the specific Receiver Enable bit (SCON0.4 or SCON1.4).
4. A falling edge is detected on the specific RX pin.

The switchback feature also works in conjunction with the transmit function. If the appropriate conditions are met, a device operating in a PMM will automatically return to divide by 4 mode when a serial port buffer (SBUF0;99h or SBUF1;C1h) is loaded. This removes the need for the user to manually set the speed to divide by 4 before initiating the transmission. The transmitter interrupt can be used to signal when the transmission is complete so that software can return the device to the appropriate PMM. Three conditions must be met for a serial port transmission to cause a switchback:

1. The device must currently be in PMM1 or PMM2.
2. The SWB bit (PMR.5) must be set.
3. A serial port transmission must be initiated by loading the specific serial port buffer (SBUF0;99h or SBUF1;C1h).

Although both the serial port transmit and receive functions are possible in PMM, it is not possible to configure the baud rate generator to any standard rate (300, 1200, 2400, etc.) in these modes, making it impossible to communicate with a standard peripheral. The use of the switchback feature is strongly recommended if serial port activity and PMM are to be used in a design.

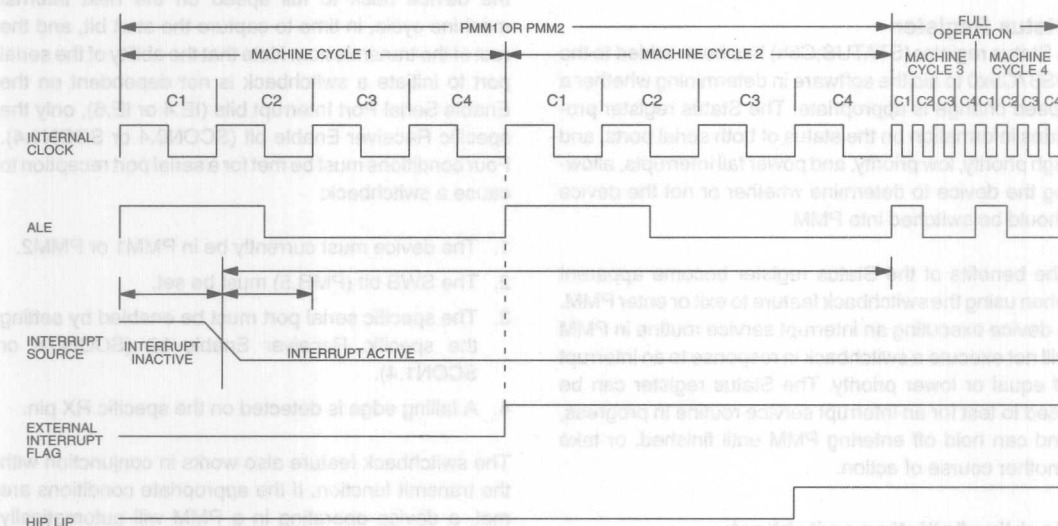
## CONSIDERATIONS WHEN USING SWITCHBACK

### Switchback Timing

One of the primary considerations when using the switchback procedure is the time required to return the

device to full speed from a PMM. This is a factor in calculating the latency associated with servicing an interrupt. Switchbacks will occur at the C1 cycle of the first instruction following the event initiating the switchback. If the current instruction in progress is a write to the IE, IP, EIP or EIE register, interrupt processing will be delayed until the completion of the following instruction. Figure 3 demonstrates the timing relationship between interrupts and switchbacks during a two-cycle instruction.

**INTERRUPT-DRIVEN SWITCHBACK Figure 3**



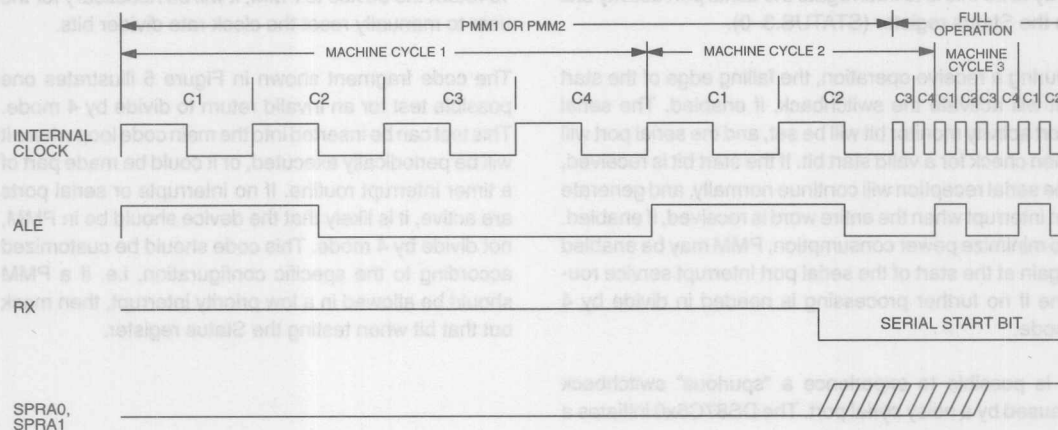
### NOTES:

1. Internal clock cycle is 16 external clock cycles for PMM1 and 256 external clock cycles for PMM2.
2. Polarity of interrupt varies with respect to external interrupt number.
3. Example shows two-cycle instruction. Execution of the interrupt and switchback will occur at the end of the last machine cycle of the instruction in which the interrupt is acknowledged.

One exception to the above timing relationship is that serial port switchbacks will occur immediately upon receipt of a falling edge on an enabled serial port receiver. The switchback will occur at the start of the

next internal clock cycle following the falling edge. Figure 4 demonstrates the timing relationship between serial port activity and switchbacks.

**SERIAL PORT DRIVEN SWITCHBACK** Figure 4



#### NOTES:

1. Internal Cx cycle is 16 external clock cycles for PMM1 and 256 external clock cycles for PMM2.
2. SPRA0 and SPRA1 will change within 1 machine cycle of the falling edge RX.
3. Example shows single-cycle instructions. Execution of the interrupt and switchback will occur at the end of the last machine cycle of the instruction in which the interrupt is acknowledged.

#### Interrupt Priority

Because the switchback feature uses interrupts to qualify execution, it is affected by interrupt priorities. The external interrupts initiate a switchback upon the start of the interrupt service routine. If a higher priority interrupt is in progress, the associated switchback will remain pending. It is not possible to enable or disable the switchback function for individual interrupt sources, except by enabling or disabling the specific interrupt.

The following example will illustrate how a problem could occur if the priority of the interrupt sources is not taken into account. Assume that the user is employing both Timer 0 and External Interrupt 1 in a design which utilizes PMM. While operating in PMM, a Timer 0 interrupt occurs, and the device begins executing the interrupt service routine (ISR). During the Timer 0 ISR, an External Interrupt 1 occurs, signaling an external event

that needs to be serviced quickly. Because both interrupts have the same priority, External Interrupt 1 will remain pending until completion of the Timer 0 ISR. Although such interrupt priorities are a normal consideration in any design, the reduced operating speed in PMM will further increase the latency associated with servicing External Interrupt 1. This could be avoided by specifying External Interrupt 1 as a high priority interrupt and leaving Timer 0 as a low priority interrupt.

A serial port-initiated switchback does not utilize the interrupt structure, and is therefore not affected by interrupt priorities. Serial port-initiated switchbacks are enabled or disabled via the specific receiver enable bit (SCON0.4 or SCON1.4). The ability of a serial port to initiate a switchback is not dependent on the Enable Serial Port Interrupt bits (IE.4 or IE.6).

clock frequency of the DS87C5x0, timing-dependent peripherals such as the serial ports can be affected. The user must be sure that the serial ports are not receiving or transmitting when switching into PMM. The simplest way to do this is to interrogate the serial port activity bits in the Status register (STATUS.3-0).

During a receive operation, the falling edge of the start bit will activate the switchback, if enabled. The serial port activity monitor bit will be set, and the serial port will then check for a valid start bit. If the start bit is received, the serial reception will continue normally, and generate an interrupt when the entire word is received, if enabled. To minimize power consumption, PMM may be enabled again at the start of the serial port interrupt service routine if no further processing is needed in divide by 4 mode.

It is possible to experience a "spurious" switchback caused by a noisy serial port. The DS87C5x0 initiates a

received, the system will abort the serial activity, clearing the activity bit, and no serial port interrupt will be executed. The switchback has already been initiated, however, and the device is now operating at full speed. To return the device to PMM, it will be necessary for the user to manually reset the clock rate divider bits.

The code fragment shown in Figure 5 illustrates one possible test for an invalid return to divide by 4 mode. This test can be inserted into the main code loop where it will be periodically executed, or it could be made part of a timer interrupt routine. If no interrupts or serial ports are active, it is likely that the device should be in PMM, not divide by 4 mode. This code should be customized according to the specific configuration, i.e. if a PMM should be allowed in a low priority interrupt, then mask out that bit when testing the Status register.

#### INVALID SWITCHBACK TEST EXAMPLE Figure 5

```

MODETEST:  PUSH  A           ;Save the current value of the accumulator.
            MOV   A, PMR      ;Move the data to a bit-addressable register.
            JB    E7, PMM_ON   ;If bit 7 is set, device is already in PMM.
CHK_STAT:  MOV   A, STATUS     ;Check status register for active interrupts.
            AND   A, #0EFh     ;Check for user-defined activity.
            JNZ   CHK_STAT     ;If activity, loop until complete.
            ; (Code can either loop until the condition
            ; clears or abort attempt to reenter PMM.)
ENA_PMM:   OR    PMR, #0C0h    ;Status okay for return to PMM. Set Clock Rate
            ; Divider bits (example shows return to PMM2)
PMM_ON:    POP   A           ;Restore accumulator.

```

#### Multiprocessor Communications in PMM

The effectiveness of PMM and the switchback feature is affected if multiprocessor communications protocols are used. The DS87C5x0 includes features that will support multiple processors on the same serial port. In serial port modes 2 and 3 it is possible to use the SM2 flag (SCON0.5 or SCON1.5) to signify that the received byte is an address. The slave address recognition registers (SADDR0;A9h, SADDR1;AAh, SADEN0;B9h, SADEN1;BAh) can be programmed to ignore a transmission (not cause a receiver interrupt) when a received address does not match a user defined pattern.

The implication of multiprocessor communications for power management is that a switchback is generated by

the detection of the first falling edge on a serial port, not the generation of a valid interrupt. As a result, an invalid address which should be ignored by a particular processor will still generate a switchback. Normally, the part could be returned to PMM at the start of the serial port interrupt service routine. Unfortunately, in the above mentioned case no interrupt will be generated. To alleviate this problem, one should avoid using a multiprocessor communication scheme in conjunction with PMM. If the system power considerations will allow for an occasional erroneous switchback, the polling scheme shown in Figure 5 can be used to place the device back into PMM.



## CLOCK SOURCE CONTROL

### Description

The DS87C5x0 incorporates a number of features to control the clock source to the device. By controlling the source of the system clock, the system designer can simultaneously achieve higher performance and decreased power consumption. To provide maximum flexibility, the DS87C5x0 will operate from three clock sources:

1. External Crystal (using internal crystal oscillator),
2. External Clock Oscillator,
3. Internal Ring Oscillator.

### EXTERNAL CRYSTAL

The most common clocking source for the DS87C5x0 is an external crystal. The DS87C5x0 incorporates a crystal amplifier which is designed to drive industry standard crystals over the operating range of the device. External crystals provide a highly accurate clock source for timing-dependent peripherals such as internal timers and serial ports, as well as device operation. The DS87C5x0 requires a fundamental-mode, parallel-resonant (also called anti-resonant) AT cut crystal. Crystal oscillators have significant start up times, however, which can delay the operation of the device when powering on or resuming from the Stop mode. The DS87C5x0 must start operation with an external crystal or external clock source after a power-on reset.

### EXTERNAL CLOCK SOURCE

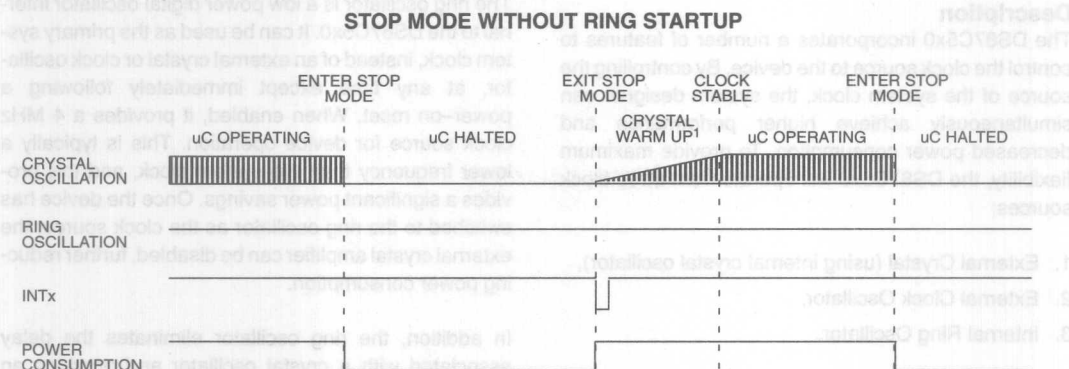
If a clock oscillator is already present in the system, it can be used as the external clock source for the DS87C5x0. External clock oscillators suffer from the same start-up delays as the internal crystal oscillator, and do not provide any special benefits. Either an external crystal or external clock source can be used to clock the device following a power-on reset or power-fail reset.

## RING OSCILLATOR

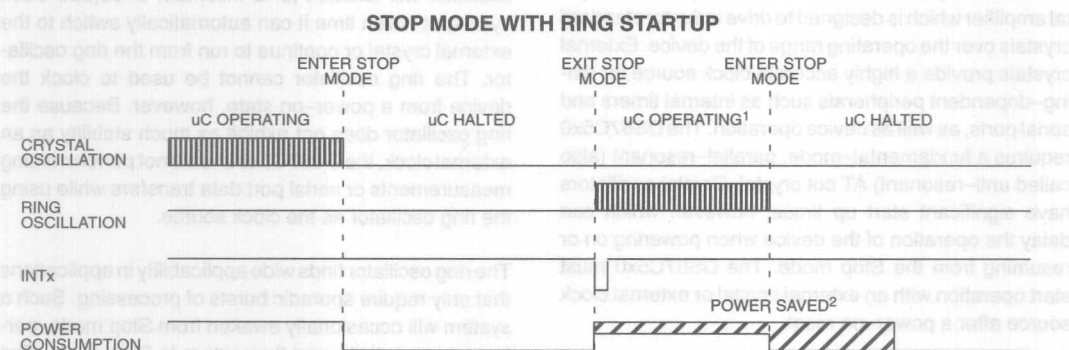
The ring oscillator is a low power digital oscillator internal to the DS87C5x0. It can be used as the primary system clock, instead of an external crystal or clock oscillator, at any time except immediately following a power-on reset. When enabled, it provides a 4 MHz clock source for device operation. This is typically a lower frequency than the system clock, and this provides a significant power savings. Once the device has switched to the ring oscillator as the clock source, the external crystal amplifier can be disabled, further reducing power consumption.

In addition, the ring oscillator eliminates the delay associated with a crystal oscillator and provides an almost instantaneous startup from Stop mode. When used to restart the DS87C5x0 from Stop mode, the ring oscillator will function for a minimum of 65,536 clock cycles, at which time it can automatically switch to the external crystal or continue to run from the ring oscillator. The ring oscillator cannot be used to clock the device from a power-on state, however. Because the ring oscillator does not exhibit as much stability as an external clock, the DS87C5x0 should not perform timing measurements or serial port data transfers while using the ring oscillator as the clock source.

The ring oscillator finds wide applicability in applications that only require sporadic bursts of processing. Such a system will occasionally awaken from Stop mode, perform some activity, and then return to Stop mode. The ring oscillator allows the system to quickly switch from the lowest power state, perform an operation, and then return to a low power state, without restarting a halted external crystal. Figure 6 shows the advantages of restarting from Stop mode with and without the ring oscillator.

**ADVANTAGES OF RING OSCILLATOR** Figure 6

1. Crystal Warm up period is 4 to 10 ms depending on crystal type and speed.



1. Diagram assumes that the operation following Stop requires less than 18 ms to complete.
2. Additional power savings due to decreased ring oscillation current compared to crystal amplifier.

Even if timing-dependent functions are necessary shortly after resuming Stop mode, the ring oscillator may be beneficial. Typically there is some processing that is necessary along with a timing routine or serial port transmission. Prior to executing the Stop command, the device should switch to the crystal as the clock source and set the RGSL bit. Upon resuming from Stop, the device can execute code while running from the ring oscillator in preparation for the timing dependent operation. The device can then loop until the RGMD bit has been cleared, indicating that the crystal

or external clock source is now the clock source, and timing dependent operations can begin.

**CLOCK CONTROL BITS**

There are a number of bits that are used to configure the clock management modes of the DS87C5x0. These allow the system to switch between different clock sources, indicate the current status of the clock source, and select how the device will resume from Stop mode. The pertinent bits are shown in Table 3.

**CLOCK CONTROL AND STATUS BIT SUMMARY** Table 3

BIT NAME	LOCATION	FUNCTION	RESET	WRITE ACCESS
XT/ $\overline{\text{RG}}$	EXIF.3	Crystal/Ring Clock Source Select. 0 = Select ring oscillator as clock source. 1 = Select crystal or external clock as clock source.	1	0 anytime; 1 when XTUP=1 and XTOFF=0
RGMD	EXIF.2	Ring Oscillator Mode Status. 0 = Crystal or external clock is current clock source. 1 = Ring oscillator is current clock source.	0	None
RGSL	EXIF.1	Ring Oscillator Select, Stop Mode. 0 = Crystal or external clock will be the clock source when resuming from Stop mode. 1 = Ring oscillator will be the clock source when resuming from Stop mode.  Note: Upon completion of crystal warm up period, device will switch to clock source designated by XT/ $\overline{\text{RG}}$ bit.	Unchanged except after power-on reset, when it is cleared to 0.	Unrestricted
XTOFF	PMR.3	Crystal Oscillator Disable. 0 = Crystal oscillator is enabled. 1 = Crystal oscillator is disabled. Device is operating from ring oscillator.	0	0 anytime; 1 when XT/ $\overline{\text{RG}}$ =0
XTUP	STATUS.4	Crystal Oscillator Warm Up Status. 0 = Oscillator warm up still in progress. 1 = Oscillator warm up complete.	1	None

**CRYSTAL OSCILLATOR STARTUP DELAY**

When power is applied to a crystal oscillator after a period of non-operation, a short period of time is required before the amplitude of the pulse is sufficient to provide a stable clock source. This can result in missed or corrupted clock signals, possibly disrupting processor operation. To ensure a valid clock signal, the DS87C5x0 uses a crystal startup counter to detect 65,536 oscillations of the external crystal or clock oscillator before allowing the device to resume operation. This means that devices utilizing slower crystals will have longer crystal startup times. The crystal startup counter is more sensitive than the internal clock circuitry, and uses the count of both bad and good pulses to determine the warm-up period. The counter value was chosen to allow the majority of crystals enough time to stabilize before releasing the device to run off the external crystal. The counter is reset anytime the XTOFF bit is cleared.

The status of the crystal startup counter can be determined by reading the Crystal Oscillator Warm Up Status Bit, XTUP (STATUS.4). Note that this bit will always be set upon a power-on reset, because the counter must

time out before the device will resume operation. For the same reason, this bit will also be set when resuming from Stop mode with the XT/ $\overline{\text{RG}}$  bit set to 1. When switching from the ring oscillator to the crystal oscillator, the XTUP bit can be used to tell when the crystal has stabilized. Attempts to switch to the external crystal before the XTUP bit has been set will be disregarded.

**SWITCHING BETWEEN CLOCK SOURCES**

On occasion the device may wish to switch between the ring oscillator and crystal oscillator. The device can switch to the ring oscillator at any time as there is no start up delay associated with the ring oscillator. Clearing the Crystal Oscillator/Ring Oscillator Select Bit, XT/ $\overline{\text{RG}}$  (EXIF.3) will enable the ring oscillator. If there is no expectation that the crystal oscillator will be needed soon, the crystal oscillator can be disabled by setting the Crystal Oscillator Disable Bit, XTOFF (PMR.3). This will provide a significant power savings. Note that clearing the XT/ $\overline{\text{RG}}$  bit does not automatically disable the crystal amplifier.

delays inherent in the external crystal. The procedure is as follows:

1. Clear the Crystal Oscillator Disable Bit, XTOFF (PMR.3) to restart the crystal oscillator.
2. Wait for the Crystal Oscillator Warm Up Status bit, XTUP (STATUS.4) to be set, indicating that the external crystal warm up period is complete.
3. Set the Crystal Oscillator/Ring Oscillator Select Bit, XT/RG (EXIF.3) to select the crystal as the clock source.

#### CLOCK SOURCE AFTER RESET

Following a power-on reset, the RGSL bit is cleared and the XT/RG bit is set. This forces the device to operate from an external crystal or external clock source, regardless of the clock source prior to the event. The crystal startup counter will be reset and begin counting

In the case of external (hardware) and watchdog resets, the XT/RG bit will remain unchanged. This allows the device to continue from the same clock source that was active before the reset event. Regardless of the state of the XT/RG bit, the XTOFF bit is cleared following any reset, which begins the crystal oscillator warm up. If the crystal will not be used, the appropriate reset routines should set the XTOFF bit to disable the crystal oscillator to conserve power.

#### CLOCK SOURCE AFTER STOP

During Stop mode, internal clocking to the DS87C5x0 is halted. Upon receipt of an external interrupt or reset, the device will use the state of the XTOFF, XT/RG, and RGSL bits prior to entering Stop mode to determine the state of the ring oscillator and crystal amplifier. The possible configurations are shown in Table 4.

**CLOCK SOURCE AFTER STOP MODE DETERMINATION Table 4**

XT/RG	XTOFF	RGSL	CLOCK SOURCE WHEN EXITING STOP MODE	CLOCK SOURCE AFTER CRYSTAL WARM-UP PERIOD	STARTUP DELAY WHEN RESUMING?	CRYSTAL OSCILLATOR STATUS
0	0	x	Ring Oscillator	Ring Oscillator	No	Oscillator enabled
0	1	x	Ring Oscillator	Ring Oscillator	No	Oscillator disabled
1	0	0	Crystal Oscillator	Crystal Oscillator	Yes	Oscillator enabled
1	0	1	Ring Oscillator	Crystal Oscillator	No	Oscillator enabled

If the clock source before entering Stop mode is the ring oscillator, the device will resume operation using the ring oscillator and continue running from the ring oscillator after the crystal warm-up period. If the device enters Stop mode running from the crystal oscillator, the Ring Oscillator Select, Stop Mode bit, RGSL (EXIF.1) determines the clock source when resuming from Stop mode. Upon completion of the crystal warm-up period the device may continue to operate from the ring oscillator, or may switch to the external crystal or clock source. This is determined by the state of the XT/RG bit prior to entering Stop mode.

It should be noted that the crystal amplifier will begin its warm-up period automatically if the XT/RG bit was set prior to entering Stop mode. This will happen if the device was running from the external crystal or external oscillator, or if it was running from the ring oscillator but with the crystal amplifier still running. (Although not a logical choice, this is theoretically possible.) When resuming from the ring oscillator with the intent to continue from the ring oscillator, however, starting the crystal warm-up process is unnecessary. To prevent the crystal warm-up, make sure the device is operating from the ring oscillator and the XTOFF bit is set before entering Stop mode.



The ring oscillator is especially useful for systems which require short bursts of processing upon resuming from Stop mode. Operating from the ring allows the system to wake up, perform a short operation, and return to Stop mode in less time that it would require for an external crystal to stabilize. This provides a two-fold power savings: The time out of Stop mode is reduced due to the quick start of the ring oscillator, and the ring oscillator itself typically uses less power than the crystal amplifier.

### RING OSCILLATOR CONSIDERATIONS

The ring oscillator used in the High-Speed Microcontroller Family is essentially a chain of inverters with a propagation delay. Although it exhibits fast start up times, it does not carry the stability of a piezo electric quartz crystal oscillator. The ring oscillator will oscillate from 3 to 4 MHz over the temperature and voltage range specified for the device. This variation makes it difficult to generate a stable time base for timers and timing-sensitive operations. Interrupt latencies will also be more difficult to calculate due to the variation of the main system clock.

It is not advised to operate the serial ports in asynchronous mode (Modes 1, 2, or 3) while running from the ring oscillator. The serial ports use internal timers to generate their baud rate, and the resulting frequency is not stable enough to support an asynchronous serial transmission. Synchronous serial transmissions in mode 0 are possible, however, due to the synchronizing clock generated by the host processor.

The use of the ring oscillator does not impair the operation of the real-time clock, watchdog timer, or Timed Access operations. The real time clock incorporated in the DS87C530 is excited by an external 32 KHz crystal which is independent of the system clock. The watchdog timer and Timed Access procedures both function with respect to internal clock cycles, not an absolute time reference, and will operate properly. If an absolute time period is required for the watchdog timer, then an external clock source is recommended.

### PERFORMING A "RING-OSCILLATOR SWITCHBACK"

The switchback feature of the DS87C5x0 allows the device to "wake up" for serial port operations when operating in PMM1 or PMM2. Although the device will execute a switchback regardless of the clock source, the device must be operating from a crystal or external clock source for the serial operation to be successful. In most cases, this would preclude the use of the ring oscillator or Stop mode if serial port operations are expected. However, it is possible to "switchback" from the ring oscillator to the crystal upon receipt of a serial transmission.

In the case of a serial transmission, the ring oscillator presents little problem; the system can simply enable the crystal oscillator, wait for the crystal to stabilize, and then begin the transmission. Serial receptions are more difficult. There is no way that a DS87C5x0 operating from a ring oscillator will be able to successfully capture a serial data transfer on the first try. One possible solution would be to employ an a handshaking protocol to confirm that the receiver is ready and the data should be resent. The key in such a scheme is to have the DS87C5x0 detect that a serial operation has been attempted and execute a section of code that will switch over to the crystal source.

The recommended approach utilizes an external interrupt as a serial port activity monitor. If a negative edge triggered interrupt such as  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ,  $\overline{\text{INT3}}$ , or  $\overline{\text{INT5}}$  is tied to the RX pin, the falling edge of a start bit will generate an interrupt and a switchback. The interrupt service routine will enable the crystal clock source and wait until it is stable, at which time the device will transmit a ready signal back to the originator. The following code example demonstrates one way to do this.

In addition, the delay associated with restarting the crystal can be avoided by keeping the crystal amplifier enabled when running from the ring oscillator. This seems counterproductive at first, as it increases the power consumption slightly when compared to running from the ring oscillator alone. However, when operating the device from a relatively high-speed crystal, the reduced speed of the ring oscillator still results in a net power savings.



**PROGRAM EXAMPLE: SOFTWARE "RING OSCILLATOR SWITCHBACK"**

```

;Program RING_SWB
;
;This program shows how the serial ports can be operated in conjunction with
;the ring oscillator. When receiving a byte, the falling edge of the start
;bit will generate an INT1. The INT1 ISR will restart the crystal oscillator.
;When the crystal has stabilized, the system will transmit a ready character
;to the originator to indicate that the receiver is ready.
;
;The core of the program continuously scans Port 1 and records the maximum
;value. The program performs two operations: transmit the maximum
;recorded value, and reset the maximum value to 0. Receipt of an invalid
;command will cause the device to return an error code to the host and
;switch back to the ring oscillator. Invalid codes can also be used to
;intentionally return the device to the ring oscillator at the completion
;of the data transfer. Valid commands will be echoed back to the originator
;to confirm receipt.
;*****
;Register Equates
P0      equ      80h      ;Port 0
SP       equ      81h      ;Stack Pointer
PCON     equ      87h      ;Power Control Register
TMOD     equ      89h      ;Timer Mode Control Register
TH1      equ      8Dh      ;Timer 1 MSB (used for baud rate generation)
P1       equ      90h      ;Port 1
EXIF      equ      91h      ;External Interrupt Flag Register
SCON0     equ      98h      ;Serial Port 0 Control Register
SBUF0     equ      99h      ;Serial Port 0 Data Buffer
IE        equ      0A8h     ;Interrupt Enable Register
P3        equ      0B0h     ;Port 3
PMR       equ      0C4h     ;Power Management Register
STATUS    equ      0C5h     ;Status Register
ACC       equ      0E0h     ;Accumulator
;R0 : Command Register.
;R1 : Maximum Value observed on Port 1.

;Bit Equates
RI_0      equ      98h      ;Serial Port 0 Receiver Interrupt Flag
TI_0      equ      99h      ;Serial Port 0 Transmitter Interrupt Flag
IE1       equ      8Ah      ;Interrupt 1 Flag.
TR1       equ      8Eh      ;Timer 1 Run control.
REN_0     equ      9Ch      ;Serial Port 0 Receiver Enable
EX1       equ      0AAh     ;External Interrupt 1 Enable
EA        equ      0AFh     ;Global Interrupt Enable

;String Equates
RDY_CHAR  equ      '!'
ERR_CHAR  equ      '?'

;Interrupt Vector Table.
                cseg at 0      ;Reset vector.

```

```

        ljmp START
        cseg at 13h          ;External Interrupt 1 vector.
        ljmp EXT_INT1
        cseg at 23h          ;Serial Interrupt 0 vector.
        ljmp SER0_INT
        cseg at 100h         ;Beginning of code segment.

START:   MOV     SP, #40h      ;Initialize stack pointer.
        MOV     P3, #0Fh      ;Set port pins as inputs.
        MOV     P1, #0FFh     ;Set port pins as inputs.
        CALL    RING_ENA      ;Switch to ring oscillator to conserve power.

        MOV     TH1, #0FDh    ;Set timer for 19200 baud rate at 11.059 MHz
        MOV     TMOD, #20h     ;Set Timer as mode 2 for baud rate generation.
        MOV     SCON0, #50h    ;Select Mode 1, enable receiver.
        ORL     PCON, #80h     ;Set SMOD for 19200 operation.
        SETB    TR1           ;Start Timer 1 for baud rate generation.

        MOV     IE, #94h      ;Enable global, serial 0, and ext. interrupt 1.
        MOV     R1, #0        ;Reset maximum value counter.

CLR_BUF: MOV     R0, #0        ;This is the reentry point after command
        ; completion that clears the command buffer.
        ; It then falls through to the main prog loop.

MAIN:    CJNE    R0, #0, COMMAND ;If R0<>0, then service pending command.

PORTSCAN: MOV    A, P1        ;Get the current port value.
        PUSH    ACC           ;Make a temporary copy of port value.
        CLR     C             ;Compare current value to maximum. If smaller,
        SUBB    A, R1         ; or equal, loop back for next check.
        POP     ACC           ;Restore port value. Note that this does not
        ; affect the carry flag from the SUBB inst.
        JC      MAIN          ;If negative number from SUBB, value is not
        ; a new maximum, so go on.
        MOV     R1, A         ;We have a new maximum. Store it.
        JMP     MAIN          ;End of main program loop

COMMAND: CJNE    R0, #'1', CHECK_2 ;If command is not XMIT_MAX, go on.

XMIT_MAX: MOV    A, STATUS     ;Host is requesting maximum value. Wait until
        JB      ACC.1, XMIT_MAX ; serial port transmit activity is complete.
        MOV     SBUF0, R1      ;Send maximum value back to host.
        JMP     CLR_BUF        ;Return to main loop to await next command.

CHECK_2: CJNE    R0, #'2', INVALID ;If command is not RESET_MAX, then an
        ; invalid command has been received.

RESET_MAX: MOV    R1, #0       ;Host is requesting that maximum value
        JMP     CLR_BUF        ; be reset. Zero value and return

```

```

; to mail loop to await next command.

INVALID:  MOV     SBUF0, #ERR_CHAR ;An invalid command has been received.
          CALL    RING_ENA        ;Return to ring oscillator and clear the
          JMP     CLR_BUF         ; command buffer. This will also be called
                                   ; intentionally by the originator to return
                                   ; the device to the ring oscillator.

;*****
;SERO_INT - This ISR handles serial port 0 interrupts. Receiver interrupts
;           will be caused by receipt of a command byte from the originator.
;           The byte will then be echoed back to confirm receipt.
;
;           Transmitter interrupts are called by the transmission of data
;           or a status character to the originator.
;*****
SERO_INT:  JB      TI_0, XMIT_INT ;Determine source of interrupt.
          MOV      R0, SBUF0      ;Interrupt was serial reception. Save command
          CLR      RI_0           ; byte in R0 for main program loop and clear
                                   ; receiver interrupt flag.
          MOV      SBUF0, R0      ;Echo data to acknowledge receipt.
          RETI
XMIT_INT:  CLR      TI_0          ;Interrupt was caused by transmitter. Clear
          RETI                   ; interrupt flag and return.

;*****
;EXT_INT1 - This ISR restarts the crystal in response to a falling edge
;           on the INT0 pin, which is also tied to the serial port 0 RX pin.
;           When crystal has stabilized, it sends a ready signal to the originator.
;           Further INT1s are disabled until the data has been received to prohibit
;           data bits from being mistaken as start bits.
;*****
EXT_INT1:  CLR      EX1           ;Disable any more serial restart interrupts
          CALL     XTAL_ENA       ;Start crystal oscillator
          MOV      SBUF0, #RDY_CHAR ;Send Ready signal
          CLR      RI_0           ;Any serial port receiver interrupts at this
                                   ; point are erroneous, so ignore them.
          RETI

;*****
;XTAL_ENA - This subroutine checks to see if the crystal is running, and if
;           not, enables it and waits until it has stabilized.
;*****
XTAL_ENA:  PUSH     ACC           ;Save accumulator
          ANL      PMR, #0F7h    ;Clear XTOFF bit to restart crystal.
XTALWAIT:  MOV      A, STATUS     ;Check XTUP: Loop until crystal has stabilized.
          JNB      ACC.4, XTALWAIT

```

```

        ORL     EXIF, #08h      ;Switch to crystal as clock source.
        SETB   REN_0          ;Crystal is active, enable receiver.
        POP    ACC            ;Restore Accumulator.
        RET                     ;Device is now running from crystal. Exit.

;*****
;RING_ENA - This subroutine checks to see if there is any serial port
;           activity, and if not, switches back to the ring oscillator.
;*****
RING_ENA:  PUSH   ACC          ;Save accumulator

WAIT_SERIAL:
        MOV     A, STATUS      ;Test lower nibble of status reg for serial
        ANL     A, #0Fh        ; port activity. If serial ports are still
        JNZ     WAIT_SERIAL    ; active, wait before switching to ring.

        CLR     REN_0          ;Ignore any serial port activity while running
                                ; from ring oscillator.
        CLR     IE1            ;Clear any outstanding serial interrupts that
        SETB    EX1            ; may have been generated by serial data and
                                ; reenable external interrupt 1 to detect the
                                ; start of another serial transfer.

        ANL     EXIF, #0F7h     ;Clear XT/RG to enable ring oscillator.
        ORL     PMR, #08h      ;Set XTOFF bit to disable crystal.

        POP     ACC            ;Restore Accumulator.
        RET                     ;Device is now running from ring. Exit.

```

## DEVELOPING A POWER MANAGEMENT FRAMEWORK

The Dallas Semiconductor approach to power management allows system designers to reduce power consumption while maintaining maximum performance. To achieve the maximum possible savings, the device operating conditions should be carefully analyzed and a power management scheme developed.

The determination of which power management modes to utilize, when to switch modes, and how to handle high-priority tasks is application dependent. No single approach will suit all possible combinations. In general, the selection of clock speeds and sources depends on the assigned tasks, and the need for timing-dependent operations such as serial ports activity.

There are two basic classes of systems which employ power management. The first is systems which hibernate or spend almost all of their operating time in a standby state, such as Stop or PMM2. These systems are often used in unattended systems to gather data or as environmental monitors. They are characterized by relatively infrequent I/O activity at specific intervals. The second class of systems usually performs a high rate of I/O activity on several devices, or otherwise must be operating constantly. A hibernation approach would be impractical in this instance as the device would spend most of its time simply restarting from the low power state. These two approaches are discussed in more detail in the following sections.

## BURST MODE OPERATION

A common mode of operation is to have the device operate in a low power state, perform a brief task, and then place the device back into a low power state until another event occurs. Actions such as a keypad press, or card reader activity fall within this category. Such peripherals typically generate an external interrupt which executes a switchback or resume from Stop mode.

The decision on what to designate as the standby state depends on the type of activity that will initiate a return to the active state. If serial port activity is expected, then the standby state must be one that can receive serial data, such as PMM2. A system which can tolerate longer interrupt latencies can use Stop mode as the low power state.

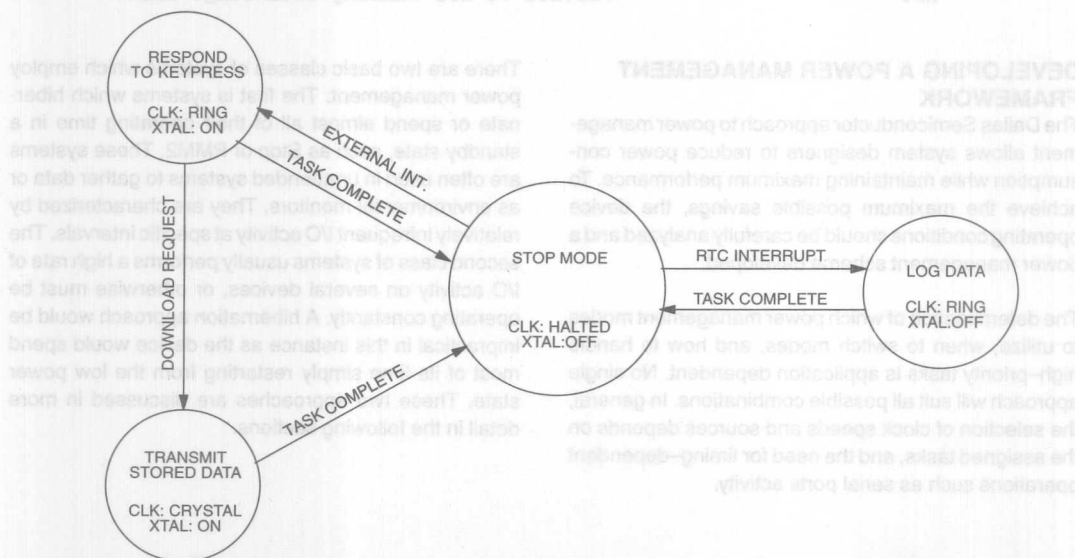
An examination of the power consumption in the various modes shows that when using a burst-mode approach, the most power savings can be gained by operating in divide by 4 mode, rather than the PMMs. Divide by 4 mode gives 16 times the performance of PMM1, but consumes only four times the current. The higher power/performance ratio of divide by 4 mode means that

less total energy will be consumed during the sub-routine. As a result, routines that wake from Stop mode to perform short bursts of activity and then return to Stop should do so in divide by 4 mode.

## PROGRAM EXAMPLE: REMOTE DATA LOGGER

The following program illustrates a generic scheme for running a remote, battery-powered data-logging device which requires only sporadic use. In this example, a DS87C530 "hibernates" in Stop mode until a keypad is pressed and performs some operation. After the operation is performed, the device will return to Stop mode. This device runs off the ring oscillator during most of its awake state, unless a serial transfer is expected or in progress. Periodically a real-time clock interrupt will cause the device to acquire data from an external source, record it in the on-chip SRAM, and return to Stop mode. Figure 7 shows the state diagram for how the device works. Although this example uses the internal real time clock of the DS87C530, it can be easily modified to work with an internal timer in PMM2 or external real time clock.

REMOTE DATA LOGGER EXAMPLE STATE DIAGRAM Figure 7





## PROGRAM EXAMPLE: BURST-MODE DATA LOGGER

```

;*****
;Program DATA_LOG
;
;This program demonstrates burst mode operation in a data logger device. The
;device remains in Stop mode until a real time clock or external interrupt
;resumes operation. When an operation is complete, the device will switch
;back to the ring oscillator and return to Stop mode.
;
;The real time clock will interrupt the system twice per hour to read a value
;from port 1, which it will store until requested. The interrupt is called
;on the hour to start the data acquisition. The routine presented is simple and
;generic, so the data could be from a D/A converter or a temperature sensor,
;for example.
;
;Upon receiving an external interrupt, the device will start the crystal
;amplifier in expectation of possible serial port activity, but continue to
;operate from the ring oscillator. If serial activity is detected or desired,
;the device will hold operation until the crystal has warmed up, and then
;switch operation to it.
;*****
;Register equate table
P0      equ    80h    ;Port 0 Latch
SP      equ    81h    ;Stack Pointer
DPL     equ    82h    ;Data Pointer 0 Low Register
DPH     equ    83h    ;Data Pointer High Register
DPL1    equ    84h    ;Data Pointer 1 Low Register
DPH1    equ    85h    ;Data Pointer High Register
DPS     equ    86h    ;Data Pointer Select Register
PCON    equ    87h    ;Power Control Register
TCON    equ    88h    ;Timer Control Register
TMOD    equ    89h    ;Timer Mode Register
TH1     equ    8Dh    ;Timer 1 MSB
P1      equ    90h    ;Port 1 Latch
EXIF    equ    91h    ;External Interrupt Flag Register
SCON0   equ    98h    ;Serial Port 0 Control Register
SBUF0   equ    99h    ;Serial Port 0 Data Buffer
P2      equ    0A0h   ;Port 2 Latch
IE      equ    0A8h   ;Interrupt Enable Register
P3      equ    0B0h   ;Port 3 Latch
IP      equ    0B8h   ;Interrupt Priority Register
PMR     equ    0C4h   ;Power Management Register
STATUS  equ    0C5h   ;Status Register
TA      equ    0C7h   ;Timed Access Register
ACC     equ    0E0h   ;Accumulator
EIE     equ    0E8h   ;Extended Interrupt Enable Register
RTASS   equ    0F2h   ;Real Time Alarm SubSecond Register
RTAS    equ    0F3h   ;Real Time Alarm Second Register
RTAM    equ    0F4h   ;Real Time Alarm Minute Register
EIP     equ    0F8h   ;Extended Interrupt Priority Register
RTCC    equ    0F9h   ;Real Time Clock Control

```

```

;Bit equate table
RI_0      equ 98h ;Serial Port 0 Receiver Interrupt Flag
TI_0      equ 99h ;Serial Port 0 Transmitter Interrupt Flag
EX1       equ 0AAh ;External Interrupt 1 Enable bit
F0        equ 0D5h ;General purpose flag.

;Constant equate table
DATA_TABLE equ 0000h ;Put data log table at start of SRAM

cseg at 0 ;Reset vector.
ljmp START
cseg at 13h ;External Interrupt 1 vector.
ljmp EXT_INT1
cseg at 23h ;Serial Interrupt 0 vector.
ljmp SER_INT0
cseg at 6Bh ;Real time clock Interrupt vector.
ljmp RTC_INT
;
cseg at 100H ;Beginning of code segment.
START:
MOV SP, #80h ;Set up stack pointer.

MOV P1, #0FFh ;Set port 1 as inputs.
MOV P3, #0Bh ;Set RXD0, TXD0 & INT1 as inputs.
MOV PMR, #01h ;Select on-chip SRAM.
MOV RTAM, #00h ;Set minute, second, and subsecond alarms.
MOV RTAS, #00h ; Alarm will wake device every hour on the hour
MOV RTASS, #00h ; to initiate temperature gathering.

MOV TA, #0AAh ;Timed access write to enable minute, second,
MOV TA, #55h ; and subsecond compares.
ORL RTCC, #0C1h

MOV SCON0, #050h ;Set serial port 0 for Mode 3
MOV TH1, #0E6h ;Timer 1 reload value for 2400 baud at 24 MHz.
MOV TMOD, #20h ;Set timer 1 to 8-bit auto reload and start it.
MOV TCON, #40h

MOV IP, #10h ;Serial port 0 and RTC high priority interrupts
MOV EIP, #20h ; so they can interrupt Ext. interrupt 1 routine.
MOV EIE, #20h ;Enable Serial port 0, Ext. Int. 1. and
MOV IE, #94h ; RTC interrupts.

;*****
;This is the main program loop. It does nothing but wait for interrupts, and
; when they are complete, switches back to the ring oscillator and puts the
; part back into Stop mode.
;*****
MAIN: ANL EXIF, #0F7h ;Switch to ring.

```

```

        ORL    EXIF, #02h    ;Enable restart from ring.
        ORL    PMR,  #08h    ;Disable crystal.
        ORL    PCON, #02h    ;Set the STOP bit to halt the device.

        JMP    MAIN          ;End of main program loop

;*****
;SER_INT0 - This ISR handles serial port 0 interrupts. Serial port interrupts
;           will only be possible when the device is "active" following a
;           keypress. The primary function of this interrupt is to transmit
;           the next character in the table until all data has been sent.
;*****
SER_INT0: JB    TI_0,XMIT_INT ;Test for transmit or receive interrupt.

        CLR    RI_0          ;This example does not receive serial data.
        RETI    ; This code is included for completeness.

XMIT_INT: CLR    TI_0          ;Clear transmit interrupt and send next byte.
        MOV    A, DPL1        ;Check to see if we are at the end of the
        CJNE   A, DPL, NOT_END ; data. If DPTR0 and DPTR1 are same, then
        MOV    A, DPH1        ; then all the data has been sent.
        CJNE   A, DPH, NOT_END
        SETB   F0             ;We have reached the end of the table.
        RETI    ; Set completion flag and exit.

NOT_END: PUSH   DPS           ;Preserve current data pointer.
        MOV    DPS, #01h      ;Switch to DPTR1 to track data pointer.
        MOVX   A, @DPTR        ;We still have data, so transmit it, restore
        MOV    SBUF0, A        ; data pointer, and return to send next byte.
        POP    DPS
        RETI

;*****
;EXT_INT1 - This ISR is generated by activity on a keypad. It causes the
;           device to read a command on Port 0 and take the appropriate action.
;           This simple example performs two functions:
;           1. Download stored data to host through serial port 0.
;           2. Clear the data table by resetting the data pointer.
;           If the command is to download stored data, it will switch to the
;           crystal first. The F0 flag is used to indicate when all the data
;           has been sent. This prevents the software from exiting the ISR
;           and reentering Stop mode before all the data has been transmitted.
;*****
EXT_INT1: ANL    PMR, #0F7h    ;Enable crystal for possible serial activity.

        MOV    A, P0           ;Read the data on Port 0 and take
        CJNE   A, #00h, CHECK_2 ; appropriate action.
        JMP    DNLOAD
CHECK_2: CJNE   A, #01h, INVALID

CLEAR:    MOV    DPTR, #DATA_TABLE ;Zero all of on-chip SRAM space from 0-3FFh.

```

```

MOV     A, #0h                ; Reset data pointer. Use A for faster fill.
NEXT_LOC: MOVX  @DPTR, A        ; Fill location & increment to next one.
INC     DPTR
MOV     R0, DPH                ; If DPH is not 04, then do next location.
CJNE    R0, #04h, NEXT_LOC

MOV     DPTR, #DATA_TABLE     ; On-chip SRAM has been cleared. Reset
RETI                                ; pointer to beginning of table and return.

DNLOAD: MOV     A, STATUS      ; Wait until crystal has stabilized.
JNB     ACC.4, DNLOAD
ORL     EXIF, #08h            ; Switch to the crystal.

MOV     A, #'!'               ; Transmit starting character. Remaining
MOV     SBUF0, A              ; data will be sent by serial port
JNB     F0, $                 ; Loop here until entire table is transmitted.
CLR     F0                    ; Transmit complete. Clear completion flag.

MOV     DPS, #01h            ; Switch to DPTR1 and reset transmit pointer.
MOV     DPTR, #DATA_TABLE
MOV     DPS, #0h              ; Switch back to DPTR0 to log data.
ANL     EXIF, #0F7h           ; Switch back to ring oscillator and return

INVALID: RETI                  ; to Stop mode.

;*****
;RTC_INT - This ISR is used to read a value from port 1. It is called every
;          30 minutes, and logs data to the data buffer. When done, it will
;          return to the main loop where it will enter Stop mode again.
;          This simple example assumes that the device will be read before
;          the data overflows the on-chip SRAM, so no error checking is
;          included.
;*****
RTC_INT: ANL     RTCC, #0FDh    ; Clear RTC Interrupt flag.
PUSH     ACC                  ; Save accumulator.
PUSH     DPS                  ; Save data pointer, in case we are interrupting
MOV     DPS, #0h              ; data download, and switch to data pointer 0.
MOV     A, P1                 ; Read data from port 1, store it in data table.
MOVX     @DPTR, A
INC     DPTR                  ; Point to next location.
XRL     RTAM, #1Eh            ; Toggles RTC alarm minute register between 0 &
                                ; 30 to interrupt on hour and half hour.
POP      DPS                  ; Restore data pointer selector and accumulator.
POP      ACC
RETI

```

## GRADUAL POWER DOWN

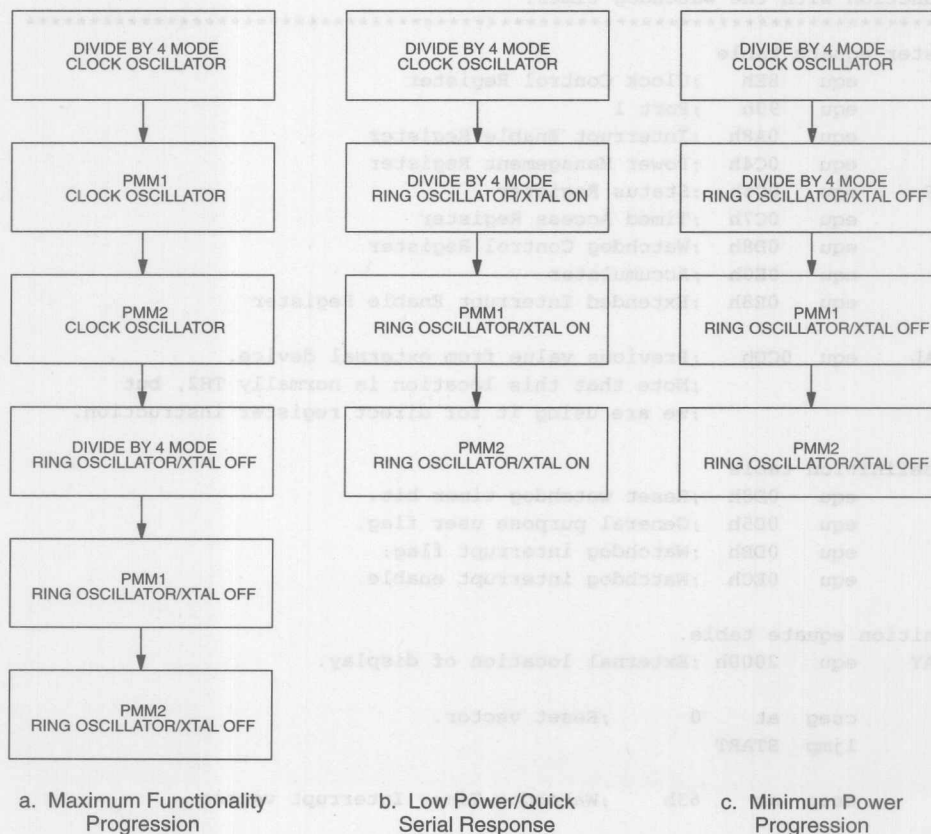
Often, a device will need to be operating constantly during periods of little activity. This may be to monitor system status, or to time critical events. While such a system can tolerate some performance degradation for the sake of power savings, it cannot halt operation by using the Stop mode. The Power Management modes used in the DS87C5x0 allow software to gradually reduce system performance based on the amount and type of tasks. This approach is similar to that used by personal computers, where inactivity for a specified period of time will cause the system to reduce its speed to the next lower level of performance.

The decision of when to switch modes, and which modes to switch between is dependent on the user's application. Constructing a "power path" is the simplest way to determine what speeds and clock source are

appropriate. For a relatively simple system, only a few states are needed. The power management modes PMM1 and PMM2 were specifically designed to be part of such a gradual power reduction.

Figure 8 demonstrates a number of power paths possible with the DS87C5x0 power management capabilities. Figure 8a shows a relatively complicated scheme which performs a gradual reduction in operating speed, while keeping the clock oscillator operating as long as possible to perform timing-dependent functions. Figure 8b switches to the ring oscillator and gradually reduces the speed of the device, but keeps the crystal amplifier enabled in case the device needs to quickly respond to serial port activity. Figure 8c provides the lowest power consumption by switching to the ring oscillator and disabling the crystal amplifier.

**SAMPLE POWER PATHS** Figure 8





**PROGRAM EXAMPLE: SYSTEM MONITOR**

The following program illustrates a basic scheme for operating a device that constantly monitors the state of a system. It operates similar to the way that a personal computer manages its power; if no activity is detected in a specified period of time it switches to the next lower power saving mode.

The program monitors an peripheral on port 1, and updates a display mapped into external memory as

needed. The watchdog timer is used to poll the status of the system, which is indicated by the F0 flag. If there is no activity before the timer times out, the device will continue to decrease its speed. Because the watchdog timer period is affected by the speed of the device, the watchdog divide ratio is adjusted to keep as constant an interval as possible. The methods demonstrated could also be used to detect a spurious switchback caused by noise on the serial port and return the device to PMM.

**PROGRAM EXAMPLE: GRADUAL POWER DOWN**

```

;*****
;Program GRADUAL.ASM
;
;This program shows a gradual version of power management. The watchdog timer
;is used to periodically check the F0 flag to see if any activity has
;occurred since the last timer interrupt. If no activity has occurred,
;then the device will switch the clock to the next lower level of operation.
;This example also demonstrates how an external interrupt would be used in
;conjunction with the watchdog timer.
;*****
;Register equate table
CKCON      equ    8Eh    ;Clock Control Register
P1         equ    90h    ;Port 1
IE         equ    0A8h   ;Interrupt Enable Register
PMR        equ    0C4h   ;Power Management Register
STATUS     equ    0C5h   ;Status Register
TA         equ    0C7h   ;Timed Access Register
WDCON      equ    0D8h   ;Watchdog Control Register
ACC        equ    0E0h   ;Accumulator
EIE        equ    0E8h   ;Extended Interrupt Enable Register

OLD_VAL    equ    0CDh   ;Previous value from external device.
                        ;Note that this location is normally TH2, but
                        ;we are using it for direct register instruction.

;Bit definition table
RWT        equ    0D8h   ;Reset watchdog timer bit.
F0         equ    0D5h   ;General purpose user flag.
WDIF       equ    0DBh   ;Watchdog interrupt flag.
EWDI       equ    0ECh   ;Watchdog interrupt enable.

;Definition equate table.
DISPLAY    equ    2000h ;External location of display.

                cseg    at    0        ;Reset vector.
                ljmp    START

                cseg    at    63h     ;Watchdog Timer Interrupt vector.

```

```

        ljmp  WDOG_INT
;
        cseg  at    100h    ;Beginning of code segment.

START:   MOV    TA, #0AAh    ;Timed access
        MOV    TA, #55h
        SETB   RWT          ;Reset watchdog timer

        MOV    DPTR, #DISPLAY ;Set data pointer to location of display.

        CLR    F0           ;Clear activity flag
        MOV    CKCON, #0C1h ;Set watchdog divide ratio to 2**26.
        MOV    EIE, #10h    ;Watchdog Interrupt Enable.
        MOV    IE, #80h     ;Enable all interrupts.
        MOV    OLD_VAL, P1

MAIN:    MOV    A, P1        ;Examine external parameter.
        CJNE   A, OLD_VAL, DIFF ;Begin display update if different.
        JMP    MAIN

DIFF:    MOV    CKCON, #0C1h ;Return watchdog divide ratio to slow speed.
        MOV    PMR, #041h    ;Switch back to divide by 4 mode.
        MOV    OLD_VAL, A    ;Save new value.
        MOVX   @DPTR, A      ;Put new value to display.
        SETB   F0           ;Set flag to indicate we have activity.
        SETB   EWDI          ;Reenable watchdog interrupt (in case device
                                ; was operating in /1024)
        JMP    MAIN          ;End of main program loop

;*****
;WDOG_INT - This ISR periodically tests for activity, and if none has been
;          detected, drops the device to the next lower clock speed.
;          Because the watchdog interval is a function of the clock divisor,
;          the watchdog divide ratio is modified in each mode to keep a
;          relatively constant interrupt frequency. The following table
;          shows the frequencies, assuming a crystal speed of 25 MHz.
;          When the device enters /1024 mode, it disables the watchdog
;          interrupt, because there is no slower speed to enter.
;
;          Clock Mode      Divide Ratio  WD1 WD0    Watchdog Timeout
;          Divide by 4      2**26         1  1      2684 mS
;          PMM1 (/64)       2**23         1  0      5368 mS
;*****
WDOG_INT: JB    F0, DONEWDOG ;If activity has been detected, do not change
                                ; speed.
        PUSH   ACC          ;Save accumulator.

        MOV    A, PMR       ;Check current speed.
        JNB    ACC.7, D4TO64 ;If CD1 = 0, then mode is /4, switch to /64.

D64TO1024: MOV    PMR, #041h ;Speed is now /64. Change to /1024 by first

```

```

MOV    PMR, #0C1h    ; going from /64 to /4 and then from /4 to /1024.
CLR    EWDI          ; Since we are now in /1024 there is no need
                    ; to go slower, so disable watchdog interrupt.

D1024:  POP    ACC      ; Restore Accumulator.
DNEWDOG: CLR    F0      ; Clear activity flag.
        MOV    TA, #0AAh ; Timed access to clear Watchdog Interrupt flag.
        MOV    TA, #55h
        CLR    WDIF
        RETI          ; Exit

D4TO64: MOV    A, STATUS ; Speed is now /4. Change to /64. Because we
        ANL    A, #0CFh   ; are entering PMM, test for activity. Check all
        JNZ    D1024      ; bits in Status Register except XTUP, and LIP
                    ; because the watchdog interrupt is low priority.
                    ; If any activity bit is set, abort speed change.
        MOV    PMR, #081h ; There is no activity. Change clock from /64.
        MOV    CKCON, #81h ; Change watchdog divide ratio from 2^26 to 2^23.
        JMP    D1024

```

### BAND-GAP DISABLING

The band-gap reference, used to detect a power failure, draws approximately 150  $\mu$ A. During Stop mode, this can be an appreciable amount of the total current drawn. The DS87C5x0 supports the option of disabling the band-gap reference, eliminating the associated current drain. When disabled, the device loses the ability to generate a power-fail interrupt or a power-fail reset. The device will continue to operate until  $V_{CC}$  drops below  $V_{RST}$ , at which time the device will cease operation. Without the bandgap reference, the device has no

way of detecting an imminent power loss, or performing an orderly shutdown. When power resumes, the device will perform a power-on reset.

Setting the Band-Gap Select bit, BGS (EXIF.0) enables the band-gap reference during Stop mode. The default or reset condition is with the bit cleared, and the band-gap disabled during Stop mode. Note that this bit can be only changed using a timed access write. It has no control of the reference during full power, PMM, or Idle modes.

# DALLAS

SEMICONDUCTOR

## Application Note 79

### Using the DS87C530 Real Time Clock

#### OVERVIEW

The DS87C530 incorporates a real time clock (RTC) and alarm to allow the user to perform real-world timing operations such as time-stamping an event, performing a task at a specific time, or executing very long timing delays. Although software timing loops or internal timers could be used for such measurements, they are crystal dependent, inefficient for long time measurements, and are incompatible with the use of power management modes. Integration of the RTC onto the DS87C530 means that only a 32.768 KHz crystal is required. No load capacitors are required with the RTC crystal. The RTC is controlled by dedicated Special Function Registers (SFRs).

The DS87C530 RTC consists of subsecond, second, minute, hour, day of the week, and two total day count registers. In addition there is an alarm register for the subsecond, seconds, minutes, and hours registers. The subsecond register provides a resolution of 1/256 of a second, and a maximum rollover count of 1 second. The registers and control bits used by the RTC are shown in Table 1. Bits and registers designated as unchanged after a reset may be indeterminate following a no-battery reset. Consult the full bit or register description for complete details.

Both user software and the internal clock directly write and read the RTC time registers (RTCSS, RTCS, RTCM, RTCH, RTCD0, RTCD1). To prevent the possibility of both user software and the internal timer accessing the same register simultaneously, the DS87C530 incorporates a register locking mechanism. Updates to the RTC time registers by the internal timer are temporarily suspended for up to 1 ms during software read or

write operations. If a subsecond timer tick should occur in the 1 ms window, it will be processed immediately as soon as either the RTCWE or RTCRE bits are cleared. To prevent the possibility of an accidental write to the RTC time registers, the RTCWE bit should be cleared as soon as the planned modifications are complete. As a protective measure, the device will clear the RTCWE bit automatically after 1 ms if it has not been cleared in software. To allow any pending timer ticks to be processed, software must wait four machine cycles between any successive modifications of the RTCWE or RTCRE bits.

This scheme will not affect the accuracy of the RTC, as any subsecond timer tick that may occur during the read or write window is only temporarily delayed, not discarded. Only the recognition of that single subsecond timer tick is delayed, and subsequent ticks will be synchronized with the clock. The only possible implication with respect to RTC operation occurs if a timer tick that would cause an alarm interrupt occurred during a time register read operation. In that case, the alarm would be delayed a fraction of a millisecond until the RTCRE bit was cleared. As mentioned, the next subsecond timer tick will occur at the proper time, so the long-term clock accuracy will not be affected.

It is critical that the 4 machine cycle setup and 1 ms window timings be observed. Any reads from the time registers before the 4 machine cycle period may return an invalid time. Writes to the time registers before the 4 machine cycle period will be ignored. Similarly, any RTC time register operations outside of the 1 ms window will result in invalid read operations or ignored write operations. For this reason, interrupts should be globally disabled before modifying any RTC register.

**REAL TIME CLOCK CONTROL AND STATUS BIT SUMMARY** Table 1

BIT NAME	LOCATION	FUNCTION	RANGE	RESET	READ/WRITE ACCESS
ERTCI	EIE.5	RTC Interrupt Enable		0	Unrestricted
PRTCI	EIP.5	RTC Interrupt Priority		0	Unrestricted
RTASS.7-0	RTASS	RTC Alarm Subsecond	0-FFh	Unchanged	Unrestricted
RTAS.5-0	RTAS	RTC Alarm Second	0-3Bh	Unchanged	Unrestricted
RTAM.5-0	RTAM	RTC Alarm Minute	0-3Bh	Unchanged	Unrestricted
RTAH.4-0	RTAH	RTC Alarm Hour	0-17H	Unchanged	Unrestricted
RTCSS.7-0	RTCSS	RTC Subsecond	0-FFh	Unchanged	Read: only if RTCRE=1. Cannot be written. Cleared when RTCWE 1 $\geq$ 0
RTCS.5-0	RTCS	RTC Second	0-3Bh	Unchanged	Read: only if RTCRE=1. Write: only if RTCWE=1. 1 ms Read/Write window
RTCM.5-0	RTCM	RTC Minute	0-3Bh	Unchanged	
RTCH.4-0	RTCH.4-0	RTC Hour	0-17h	Unchanged	
DOW2-0	RTCH.7-5	RTC Day of Week	0-7h	Unchanged	
RTCD1.7-0 RTCD0.7-0	RTCD1, (MSB) RTCD0, (LSB)	RTC Day	0-FFFFh	Unchanged	
SRCE	RTCC.7	RTC Subsecond Compare Enable		Unchanged	Unrestricted
SCE	RTCC.6	RTC Second Compare Enable		Unchanged	Unrestricted
MCE	RTCC.5	RTC Minute Compare Enable		Unchanged	Unrestricted
HCE	RTCC.4	RTC Hour Compare Enable		Unchanged	Unrestricted
RTCRE	RTCC.3	RTC Read Enable		0	Unrestricted
RTCWE	RTCC.2	RTC Write Enable		0	Read: Unrestricted Write: Timed Access
RTCIF	RTCC.1	RTC Interrupt Flag		0	Unrestricted
RTCE	RTCC.0	RTC Enable		Unchanged	Read: Unrestricted Write: Timed Access
E4K	TRIM.7	External 4096 Hz RTC Signal Enable		0	
X12/6	TRIM.6	RTC Crystal Capacitance Select		Unchanged	
TRM2-0	TRIM.5 TRIM.3 TRIM.1	RTC Trim Bit 2-0		Unchanged	Read: Unrestricted Write: Timed Access
TRM2-0	TRIM.4 TRIM.2 TRIM.0	RTC Inverted Trim Bit 2-0		Unchanged	Read: Unrestricted Write: Timed Access, must be inverse of TRM2-0



## STARTING AND STOPPING THE RTC

The operation of the RTC crystal amplifier is controlled by the RTC Enable bit, RTCE (RTCC.0). This bit can only be accessed by a Timed Access procedure, and is unaffected by any operational reset. The state of the RTCE bit is undefined after a no-battery reset, however, and should be initialized. Clearing the RTC Enable bit will halt operation of the crystal amplifier and the clock, but all register values (including the time when the clock was disabled) will be retained. This may be desirable to preserve the life of the backup energy source during periods of storage. When restarting the RTC crystal oscillator, either from a no-battery reset condition or by setting the RTC Enable bit, the crystal start-up time must be observed. There is no direct way to detect when the RTC crystal oscillator has stabilized, and the system software must allow sufficient stabilization time when restarting the RTC. Crystal startup times are specified by the crystal manufacturer, but are usually on the order of 1 second.

After a loss of battery power or when attaching a battery for the first time it will be necessary to initialize the RTC. Although there is no status bit to indicate a no-battery reset, there are several ways to detect when the real time clock has lost power / time. The best way is to monitor a reserved location in on-board memory. Because the DS87C530 on-chip SRAM contents are preserved by the same energy source as the RTC, an unexpected change in a previously loaded memory location can indicate a loss of battery power.

## READING THE TIME

Reading the current time from the RTC is accomplished by the following procedure:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Set the RTCRE bit (RTCC.3),
3. Wait 4 machine cycles,
4. Read the appropriate register(s) within 1 ms of RTCRE being set,
5. Clear the RTCRE bit (RTCC.3),
6. Enable interrupts by setting the EA bit (IE.7).

## SETTING THE TIME

The time on the DS87C530 is set by writing to the Clock Registers. The Second, Minute, Hour, Day of the Week, and Day Count can be set by writing to the respective registers. It is not possible to set the Real Time Clock

Subsecond Register (RTCS; FBh). This register is automatically reset to 00h when the RTCWE bit is cleared, either through software or the automatic time-out of the 1 ms write window. The procedure for setting an RTC time register is as follows:

1. Disable all interrupts by clearing the EA bit (IE.7),
2. Perform a Timed Access procedure,
3. Set the RTCWE bit (RTCC.2),
4. Wait 4 machine cycles,
5. Write the appropriate register(s) within 1 ms of RTCWE being set,
6. Perform a Timed Access procedure,
7. Clear the RTCWE bit (RTCC.2),
8. Enable interrupts by setting the EA bit (IE.7).

## USING THE RTC ALARM

The RTC alarm function is used to generate an interrupt when the RTC value matches selected alarm register values. An alarm can be triggered by a match on one or more of the following alarm registers: Subsecond (RTASS; F2h), Second (RTAS; F3h), Minute (RTAM; F4h), and Hour (RTAM; F5h). Note that there is no alarm register associated with the RTC Day or Day of Week Registers. If an alarm is desired on a specific date, an alarm can be executed once a day and user software can compare the current date against the Day Register. It is not necessary to set the RTC Write Enable bit when setting the alarm registers.

The alarm can be set to occur on a match with any or all of the alarm registers. An alarm can occur on a unique time of day, or a recurring alarm can be programmed every subsecond, second, minute, or hour. The specific alarm registers to be compared are selected by setting or clearing the corresponding compare enable bits (RTCC.7-4). Any compare bit which is cleared will result in that register being treated as a 'Don't Care' when evaluating alarm conditions. Clearing all the compare enable bits will disable the ability of the RTC to cause an interrupt, and will immediately clear the RTC Interrupt Flag (RTCC.1). Unlike some interrupts, the RTC flag is not cleared by exiting the RTC interrupt service routine and must be done in software.

The general procedure for setting the RTC alarm registers to cause a RTC interrupt is as follows:

1. Clear the RTC Interrupt Enable bit (EIE.5),

2. Clear all RTC Alarm Compare enable bits (ANL RTCC, #0Fh),
3. Write one or more RTC Alarm registers,
4. Set the desired RTC Alarm Compare enable bits.
5. Set the RTC Interrupt Enable bit (EIE.5).

Setting the alarm to cause an interrupt for a single time during a 24-hour period is done by setting all the alarm registers to the desired value and enabling all compare bits. For example, if an alarm was desired at 11:45:00 am, the following configuration would be used:

Alarm Subsecond (RTASS)	00 subseconds	= 00h
Alarm Second (RTAS)	00 seconds	= 00h
Alarm Minute (RTAM)	45 minutes	= 2Dh
Alarm Hour (RTAH)	11 hours	= 0Bh
Clock Control (RTCC)	subsecond compare	= F1h
	second compare	
	minute compare	
	hour compare	
	RTC enable	

A recurring alarm is enabled by disabling the compare enable bits associated with one or more alarm registers. In general, a recurring alarm is set using the next lower time increment. For example, if an alarm once an hour was desired, a compare on the RTAM Register would be performed, because the RTCM register will match RTAM register only once an hour. For example, if an alarm once an hour, on the half hour was desired, the following configuration would be used:

Alarm Subsecond (RTASS)	00 subseconds	= 00h
Alarm Second (RTAS)	00 seconds	= 00h
Alarm Minute (RTAM)	30 minutes	= 1Eh
Alarm Hour (RTAH)	11 hours	= 00h
Clock Control (RTCC)	subsecond compare	= E1h
	second compare	
	minute compare	
	RTC enable	

In the above example, the subsecond, second, and minute registers are programmed and the corresponding compare enable bits are set, even though only a match on the minute register is desired. This is because a don't care is always treated as a match for the purposes of evaluating alarms. If the SSCE and SCE bits were cleared to 0 (don't care) in the above example, then a match (and interrupt) would occur during every subsecond of the minute in which the RTAM register matched. This would cause 15,360 interrupts, which is most likely

not the desired effect. In general, when specifying a recurring alarm all the compare bits below the largest time increment should be enabled and the corresponding alarm registers loaded with 00h or a known value.

Alarms can occur synchronously when the clock rolls over to match the alarm condition or asynchronously if the alarm registers are set to a value that matches the current time. Note that only one alarm may occur per subsecond tick. This means that if a synchronous alarm has already occurred during the current subsecond, software cannot cause an asynchronous alarm in the same subsecond.

While this is a relatively minor point, it can have implications if software expects to use the asynchronous capabilities of the alarm. For example, assume an RTC interrupt occurs as when the alarm registers match the current time a 01:00:00:00 (1 hour, 0 minutes, 0 seconds, 0 subseconds). The RTC interrupt is relatively short, taking much less than one subsecond tick (<4 ms), and execution returns to the main program. Immediately upon exiting the RTC interrupt routine, an event occurs that requires software to cause an alarm on the hour by setting the alarm to match on 00 minutes, 00 seconds, 00 subseconds. Normally, setting this alarm condition with the time at 01:00:00:00 would immediately cause an RTC interrupt to occur; but because we have already had an alarm in this subseconds, the condition will not be recognized. The alarm will be missed because it will not be evaluated until the next subsecond tick, when the time will have changed to 01:00:00:01. The designer should guard against the possibility of using synchronous asynchronous alarms in the same subsecond.

Because an alarm condition can occur asynchronously, care must be exercised that a match is not accidentally enabled while writing to the alarm registers. For example, assume that the current time is 0B:00:00:00 and the current alarm conditions are 00:00:00:00. Suppose that software changes the alarm to 0B:01:00:00. If the hour, second, minute, and subsecond compare enables are enabled and the first instruction is MOV RTCH, #0B0H, an alarm will occur immediately instead of at the intended time. The best way to avoid this is to disable all compare enables before changing the RTC alarm registers.

## RTC SOFTWARE TRICKS

There are a number of simple tricks that can be used to simplify software associated with RTC operations. The 4 machine cycle delay can be performed using a CJNE A,A,\$ instruction. Compared to using 4 NOPs, this is a single instruction, and is 1 byte shorter.

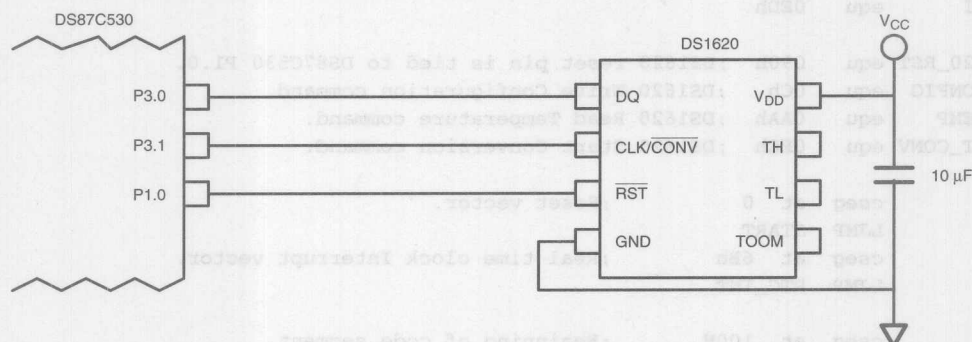
The DS87C530 RTC allows software to dynamically vary the alarm registers to achieve a wide range of intervals. Often software will want to interrupt regularly on half-increments of time (every 30 seconds, 30 minutes, etc.). This can be easily done using the XRL instruction. For example, if the RTAM register is set to 00h, the instruction XRL RTAM, #1Eh will change the contents to 1Eh. Performing the instruction again will change it back to 00h. Placing this instruction at the start of the RTC interrupt routine will cause the appropriate alarm register to be easily and quickly modified each time the interrupt is called.

a DS87C530 is awoken from Stop mode every 30 minutes to read the temperature from a Dallas Semiconductor DS1620 Digital Thermometer and Thermostat. The DS1620 is addressed via serial port 0, using serial mode 0. When the interrupt is called, the DS87C530 will use the ring oscillator to perform a fast resume from Stop, and signal the thermostat to begin a temperature conversion. It will then reset the RTC alarm to occur again in 1 second. This will allow time for the conversion and the crystal warm-up period to complete, after which the device will automatically switch back to the crystal as the clock source. The DS87C530 will read the temperature and transmit it, along with the hour and minute, back to a host system connected to serial port 1. It will then return to Stop mode to await the next alarm. Figure 1 shows a partial schematic for interfacing the DS87C530 and DS1620. If the DS1620 is to be separated from the microcontroller by a long distance, filtering may be necessary on the clock and data lines to reduce noise.

## PROGRAM EXAMPLE: DATALOGGER

The following program illustrates a generic scheme for operating a remote data logging station. In this example,

### DS1620 INTERFACE EXAMPLE Figure 1



```
;Program DATALOGR.ASM
;
;This program demonstrates how to use the RTC to periodically service an
;external device. The device halts in Stop mode until awoken by an RTC interrupt
;every half hour. It then reads the temperature from a DS1620 Digital
;Thermostat and transmits it, with a time stamp, to the host via serial port 1.
```

```
;Register equate table
SP      equ 81h ;Stack Pointer
PCON    equ 87h ;Power Control Register
TCON    equ 88h ;Timer Control Register
```

```

TMOD      equ    89h    ;Timer Mode Register
TH1       equ    8Dh    ;Timer 1 MSB
CKCON     equ    8Eh    ;Clock Control Register
P1        equ    90h    ;Port 1
EXIF      equ    91h    ;External Interrupt Flag Register
SCON0     equ    98h    ;Serial Port 0 Control Register
SBUF0     equ    99h    ;Serial Port 0 Data Buffer
P3        equ    0B0h   ;Port 3
SCON1     equ    0C0h   ;Serial Port 1 Control Register
SBUF1     equ    0C1h   ;Serial Port 1 Data Buffer
TA        equ    0C7h   ;Timed Access Register
WDCON     equ    0D8h   ;Watchdog Control Register
ACC       equ    0E0h   ;Accumulator
RTASS     equ    0F2h   ;Real Time Alarm Subsecond Register
RTAS      equ    0F3h   ;Real Time Alarm Second Register
RTAM      equ    0F4h   ;Real Time Alarm Minute Register
RTCC      equ    0F9h   ;Real Time Clock Control
RTCM      equ    0FCh   ;Real Time Clock Minute Register
RTCH      equ    0FDh   ;Real Time Clock Hour Register

```

;Bit equate table

```

RI0       equ    098h
TI0       equ    099h
REN0      equ    09Ch
EA        equ    0AFh
TI1       equ    0C1h
ERTCI     equ    0EDh

```

```

DS1620_RST equ    090h ;DS1620 reset pin is tied to DS87C530 P1.0.
WR_CONFIG  equ    0Ch  ;DS1620 Write Configuration command.
RD_TEMP    equ    0AAh ;DS1620 Read Temperature command.
START_CONV equ    0EEh ;DS1620 Start Conversion command.

```

```

cseg      at 0          ;Reset vector.
LJMP      START
cseg      at 6Bh        ;Real time clock Interrupt vector.
LJMP      RTC_INT

```

;

```

cseg      at 100H       ;Beginning of code segment.

```

```

START:    MOV     SP,    #40h    ;Initialize Stack pointer.
          MOV     EXIF,   #0Ah    ;Enable ring oscillator restart from Stop mode.
          MOV     P3,    #03h    ;Set P3.1 & P3.0 high to use serial port 0.
          MOV     SCON0, #20h    ;Set serial port mode 0, 4 tclk.

          MOV     P1,    #0Ch    ;Set P1.2 & P1.3 high to use serial port 1.
                                   ;Clear P1.0 to reset DS1620.
          MOV     SCON1, #40h    ;Set serial port mode 1.
          MOV     TMOD,   #20h    ;Configure timer 1 for 9600 baud
          MOV     TH1,    #0FDh   ; at 11.0592 MHz.
          MOV     TCON,   #40h    ;Start timer.

```

```

;Configure the DS1620
    SETB DS1620_RST    ;Remove DS1620 reset to start operation.
    MOV  A, #WR_CONFIG ;Send command to address configuration byte.
    CALL OUT_1620
    MOV  A, #03h        ;Set Configuration byte = CPU & 1-Shot Mode.
    CALL OUT_1620
    CLR  DS1620_RST    ;Assert DS1620 to end operation.

;Set up the RTC
    MOV  RTAM, #00h    ;Clear all alarm registers. Alarm will ring
    MOV  RTAS, #00h    ; on the next hour to start temperature
    MOV  RTASS, #00h   ; conversion.
    MOV  RTCC, #081h   ;Set alarms so we get a reading at start.
    SETB ERTCI        ;Enable RTC interrupt.
    SETB EA           ;Global interrupt enable.

MAIN:    ORL  PCON, #02h ;Set STOP bit to enter Stop mode.
    JMP  MAIN          ;End of main program loop. Program will return
                        ; here after RTC interrupt is complete.

;*****
;RTC_INT - This ISR reads the temp from the DS1620 and outputs the data to
;          serial port 1. The routine starts the conversion, and waits for 1
;          second to allow conversion to complete and crystal to stabilize.
;          When the conversion is complete, the device will read the temperature
;          and send the hour, minute and temperature to the host. The RTAM
;          register will be modified to alarm again in 30 minutes.
;*****
RTC_INT:  MOV  RTCC, #081h ;Clear RTCI flag and second compare enable
          ; bit to generate another alarm in 1 second.
          PUSH ACC         ;Save accumulator.

          SETB DS1620_RST ;Remove DS1620 reset to start operation.
          MOV  A, #START_CONV ;Initiate first temp conversion.
          CALL OUT_1620
          CLR  DS1620_RST ;Assert DS1620 to end operation.

          ORL  RTCC, #08h ;Enable RTC read process, and delay 4 machine
          CJNE A, ACC, $   ; cycles for time registers to stabilize.
          MOV  R7, RTCM    ;Save minute and hour so we can transmit
          MOV  R6, RTCH    ; them as soon as crystal has stabilized.
          ANL  RTCC, #0F7h ;Reenable time register updates.

WAIT:    MOV  A, RTCC      ;Wait for RTC interrupt flag to be set,
          JNB  ACC.1, WAIT ; indicating that conversion is done. The
          ; one second delay will be sufficient for the
          ; crystal to stabilize, so switch to it now.

          XRL  RTAM, #1Eh ;Change alarm to ring on next half hour.
          MOV  RTCC, #0E1h ;Clear RTCI flag, and set compare bits
          ; so next alarm will be generated in 30 min.

```



```

MOV    A, #'!'          ;Transmit start character.
CALL   OUT_HOST          ;Remove DS1620 reset to start operation.
MOV    A, R6             ;Transmit the hour.
CALL   OUT_HOST          ;Assert DS1620 to end operation.
MOV    A, R7             ;Transmit the minute.
CALL   OUT_HOST          ;Remove DS1620 reset to start operation.
SETB   DS1620_RST        ;Remove DS1620 reset to start operation.
MOV    A, #RD_TEMP       ;Conversion is done. Send command to read temp.
CALL   OUT_1620          ;Clear all alarm registers.
CALL   IN_1620           ;Read LSB of temperature and send it to host.
CALL   IN_1620           ;Read MSB of temperature and send it to host.
CLR    DS1620_RST        ;Assert DS1620 to end operation.
POP     ACC              ;Restore accumulator and go back to sleep.
RETI

;*****
;OUT_HOST - This routine sends data to the host system via serial port 1.
;*****
OUT_HOST: MOV     SBUF1, A ;Move out byte.
          JNB     TI1, $   ;Wait until data has been transmitted.
          CLR     TI1      ;Clear TI1.
          RET

;*****
;OUT_1620 - This routine sends data to the DS1620 via serial port 0.
;*****
OUT_1620: MOV     SBUF0, A ;Move out byte.
          JNB     TI0, $   ;Wait until data has been transmitted.
          CLR     TI0      ;Clear TI1.
          RET

;*****
;IN_1620 - This routine reads a byte from the DS1620 and echoes it back
;          through serial port 1.
;*****
IN_1620: SETB     REN0     ;Enable receiver to clock in data.
          JNB     RI0, $   ;Wait until data has been received.
          CLR     REN0     ;Disable receiver to prevent reception.
          CLR     RI0      ;Clear RI.

          MOV     A, SBUF0 ;Echo data through serial port 1.
          CALL    OUT_HOST ;Wait for RTC interrupt to be done.
          RET

;*****
;Change alarm to ring on next half hour
;Clear RCI flag, and set compare div
;so next alarm will be generated in 30 min

```

**PROGRAM EXAMPLE: RTC INTERFACE**

The following program is a general purpose interface routine to set the RTC and display its status. The program communicates through serial port 0, and allows

the user to set the time and date, set the alarm registers, and indicates when an alarm has occurred. For the sake of simplicity, the program inputs decimal values of time and outputs hexadecimal values.

```

;*****
;Program RTC_UTIL.ASM
;
;This program responds to commands received over the serial port to set
;and read the date, time and alarm information in the DS87C530 Real Time Clock.
;The program initializes the serial port for operation at 28800 baud with an
;11.0592 MHz clock.
;*****
;Register equate table
SP      equ    81h    ;Stack Pointer
DPL     equ    82h    ;Data pointer low register
DPH     equ    83h    ;Data pointer high register
PCON    equ    87h    ;Power Control Register
TCN     equ    88h    ;Timer Control Register
TMOD    equ    89h    ;Timer Mode Register
TH1     equ    8Dh    ;Timer 1 MSB
EXIF     equ    91h    ;External Interrupt Flag Register
SCON0   equ    98h    ;Serial Port 0 Control Register
SBUF0   equ    99h    ;Serial Port 0 Data Buffer
P3      equ    0B0h   ;Port 3
TA       equ    0C7h   ;Timed Access Register
ACC     equ    0E0h   ;Accumulator
B       equ    0F0h   ;B Register
RTASS   equ    0F2h   ;Real Time Alarm Subsecond Register
RTAS    equ    0F3h   ;Real Time Alarm Second Register
RTAM    equ    0F4h   ;Real Time Alarm Minute Register
RTAH    equ    0F5h   ;Real Time Alarm Hour Register
EIP     equ    0F8h   ;Extended Interrupt Priority Register
RTCC    equ    0F9h   ;Real Time Clock Control
RTCSS   equ    0FAh   ;Real Time Clock Subsecond register
RTCS    equ    0FBh   ;Real Time Clock Second
RTCM    equ    0FCh   ;Real Time Clock Minute
RTCH    equ    0FDh   ;Real Time Clock Hour
RTCD0   equ    0FEh   ;Real Time Clock Day Register 0
RTCD1   equ    0FFh   ;Real Time Clock Day Register 1

;Bit equate table
RIO     equ    98h    ;Serial Port 0 Receiver Interrupt Flag
TIO     equ    99h    ;Serial Port 0 Transmitter Interrupt Flag
EA      EQU    0AFh   ;Global Interrupt Enable.
ERTCI   equ    0EDh   ;Real Time Clock Interrupt Enable.

;Constant equate table
CR      equ    0Dh
LF      equ    0Ah

```

```

cseg at 0 ;Reset vector.
LJMP START
cseg at 6BH ;Real time clock Interrupt vector.
LJMP RTC_INT

*****
cseg at 100H ;Beginning of code segment.
;Data & string tables.
HEX_TABLE: DB '0123456789ABCDEF'
NEW_LINE: DB CR, LF, 0
YES: DB 'Y ', 0
NO: DB 'N ', 0
COMPARE: DB CR, LF, 'Compare enabled: ', 0
COMPARE_Q: DB ' Enable compare (Y/N)? ', 0
ALARM_MSG: DB CR, LF, 'Alarm: ', 0
TT_BANNER: DB CR, LF, CR, LF, 'DS87C530 RTC UTILITY'
DB CR, LF, ' A - Set Alarm, T -Set Time'
DB CR, LF, ' any other key to show registers'
DB CR, LF, CR, LF, 'RTC registers: ', 0
ALM_BANNER: DB CR, LF, 'Alarm register: ', 0
NEW_BANNER: DB CR, LF, CR, LF, 'Enter new alarm register settings:', 0
SET_BANNER: DB CR, LF, 'Enter new time:', 0
SS_BANNER: DB CR, LF, 'Subsecond: ', 0
S_BANNER: DB CR, LF, 'Second: ', 0
M_BANNER: DB CR, LF, 'Minute: ', 0
H_BANNER: DB CR, LF, 'Hour: ', 0
DW_BANNER: DB CR, LF, 'Day of Week: ', 0
DC_BANNER: DB CR, LF, 'Day Count: ', 0
DW_STRING: DB 'Disabled ', 0, 'Sunday ', 0, 'Monday ', 0, 'Tuesday ', 0
DB 'Wednesday ', 0, 'Thursday ', 0, 'Friday ', 0, 'Saturday ', 0

;Initialize part.
START: MOV SP, #80h ;Set up stack pointer.

MOV P3, #0Bh ;Set RXD0, TXD0 & INT1 as inputs.
MOV RTAM, #00h ;Initialize alarm registers to known values.
MOV RTAS, #00h
MOV RTASS, #00h

MOV TA, #0AAh ;Timed access write to enable RTC.
MOV TA, #55h
MOV RTCC, #01h

MOV SCON, #050h ;Set serial port 0 for Mode 1, divide by 12.
MOV TH1, #0FEh ;Timer 1 value for 28800 baud at 11.0592 MHz.
MOV TMOD, #20h ;Set timer 1 to 8-bit auto reload and start it.
MOV TCON, #40h
ORL PCON, #80h ;Set SMOD bit to get 28800 baud.

SETB ERTCI ;Enable RTC interrupts.
SETB EA

```

```

        LJMP    TELL_TIME    ;Display the time.
;*****
;This is the main program loop. It waits for a character on serial port 0,
;and then takes the appropriate action.
;*****

CHAR_TEST: JNB    RI0, $      ;Wait for incoming command character.
           CLR    RI0
           MOV    A, SBUF0    ;Test to see what to do.
CHECKT:    CJNE   A, #'T', CHECKA ;T - set time.
           LJMP   SET_TIME
CHECKA:    CJNE   A, #'A', TT_JUMP ;A - set alarm.
           LJMP   SET_ALARM
TT_JUMP:   LJMP   TELL_TIME    ;else display time.

;*****
;SET_TIME sets the current time.
;*****
SET_TIME:  MOV    DPTR, #SET_BANNER ;Display set time banner.
           CALL   OUT_STRING
           MOV    DPTR, #H_BANNER  ;Get hour & save temp copy.
           CALL   OUT_STRING
           CALL   IN_TIME
           ANL    A, #1Fh          ;Make sure day of week bits are 0.
           MOV    R4, A
           MOV    DPTR, #M_BANNER  ;Get minute & save temp copy.
           CALL   OUT_STRING
           CALL   IN_TIME
           MOV    R5, A
           MOV    DPTR, #S_BANNER  ;Get second & save temp copy.
           CALL   OUT_STRING
           CALL   IN_TIME
           MOV    R6, A
           MOV    DPTR, #DC_BANNER ;Get day count(2 bytes) & save temp copies.
           CALL   OUT_STRING
           CALL   IN_TIME
           MOV    R2, A
           CALL   IN_TIME
           MOV    R3, A
           MOV    DPTR, #DW_BANNER ;Get day of week value and add it on to
           CALL   OUT_STRING      ; the upper 3 bits of the hour register.
           CALL   IN_TIME
           SWAP   A
           RL     A
           ANL    A, #0E0h
           ORL    A, R4
           XCH    A, R4
           MOV    DPTR, #NEW_LINE  ;Add a blank line for esthetics.
           CALL   OUT_STRING

```

```

MOV     TA, #0AAh      ;We have all the values, now save them.
MOV     TA, #055h      ;Perform a timed access to write to
ORL     RTCC, #04h     ;set new time & date.
CJNE    A, ACC, $      ;Delay 4 machine cycles.
MOV     RTCSS, R7
MOV     RTCS, R6
MOV     RTCM, R5
MOV     RTCH, R4
MOV     RTCD0, R3
MOV     RTCD1, R2
MOV     TA, #0AAh      ;Clear RTCWE bit to prevent accidental
MOV     TA, #055h      ;changes to time registers.
ANL     RTCC, #0FBh

LJMP    CHAR_TEST      ;Return and wait for another event.

;*****
; TELL_TIME displays the current time, alarm registers, and alarm status.
;*****
TELL_TIME: MOV     DPTR, #TT_BANNER ;Display current time.
CALL    OUT_STRING
CALL    OUT_TIME

MOV     DPTR, #ALM_BANNER ;Display alarm registers.
CALL    OUT_STRING
MOV     R7, RTASS
MOV     R6, RTAS
MOV     R5, RTAM
MOV     R4, RTAH
CALL    DISP_TIME

MOV     DPTR, #COMPARE     ;Now display the compare bits.
CALL    OUT_STRING
MOV     A, RTCC

CALL    DISP_COMP          ;Display Hour compare bit.
RR     A
CALL    DISP_COMP          ;Display Minute compare bit.
RR     A
CALL    DISP_COMP          ;Display Second compare bit.
RR     A
CALL    DISP_COMP          ;Display Subsecond compare bit.

ORL     RTCC, #08h        ;Set the read bit to stop RTC update.
CJNE    A, ACC, $        ;Delay 4 machine cycles.
MOV     R4, RTCH          ;Read the hour register.
MOV     R3, RTCD0         ;Read the day count registers.
MOV     R2, RTCD1
ANL     RTCC, #0F7h        ;Clear the read bit to restart RTC.

```



```

MOV    DPTR, #DW_BANNER    ;Output Day of Week banner.
CALL   OUT_STRING          ; the upper 3 bits of the hour register.
MOV    A, R4               ;Day of week is stored in upper 3 bits
SWAP   A                   ; of hour register. Move it to bits 2-0
RR     A                   ; and multiply by 10 to get location
ANL    A, #07h             ; within day of week table to start.
MOV    B, #0Ah
MUL    AB
MOV    DPTR, #DW_STRING    ;Now add offset to starting address
ADD    A, DPL              ; of data table to calculate new
JNC    NO_INC              ; data pointer location.
INC    DPH
NO_INC: MOV    DPL, A
CALL   OUT_STRING

MOV    DPTR, #DC_BANNER    ;Output day count banner.
CALL   OUT_STRING
MOV    A, R2               ;Send both registers of day count.
CALL   OUT_DIGIT
MOV    A, R3
CALL   OUT_DIGIT
MOV    DPTR, #NEW_LINE     ;Add a blank line for aesthetics.
CALL   OUT_STRING

LJMP   CHAR_TEST           ;Return and wait for another event.

;This routine displays the status of the compare enable bit.
DISP_COMP: JNB    ACC.4, NO_COMP    ;Display the hour compare bit.
MOV     DPTR, #YES
JMP     OUT_COMP
NO_COMP: MOV     DPTR, #NO
OUT_COMP: CALL    OUT_STRING
RET

;This routine outputs the current time.
OUT_TIME: ORL     RTCC, #08h        ;Set the read bit to stop RTC update.
CJNE    A, ACC, $              ;Delay 4 machine cycles.
MOV     R7, RTCSS               ;Grab the current time / date and store
MOV     R6, RTCS                ; them temporarily in working registers.
MOV     R5, RTCM
MOV     R4, RTCH
ANL     RTCC, #0F7h             ;Clear the read bit to restart RTC.

DISP_TIME: MOV     A, R4          ;Output hour.
ANL     A, #01Fh                ;Mask off day of week bits.
CALL    OUT_DIGIT

MOV     A, R5                   ;Output Minute.
CALL    OUT_CDIGIT

MOV     A, R6                   ;Output second.
CALL    OUT_CDIGIT

```

```

        MOV     A, R7                ;Output subsecond.
        CALL    OUT_CDIGIT
        RET

;*****
;SET_ALARM sets the alarm registers.
;*****
SET_ALARM: CLR     ERTCI                ;Disable RTC interrupt and clear flag
           ANL     RTCC, #0Fh          ; during this section so that alarms will
                                           ; not be called while enables are changing.
           MOV     DPTR, #NEW_BANNER
           CALL    OUT_STRING
           MOV     DPTR, #H_BANNER
           CALL    OUT_STRING
           CALL    IN_TIME              ;Get hour & save temp copy.
           MOV     R4, A
           CALL    QUERY
           JNC     ASK_M
           ORL     RTCC, #10h          ;Enable hour compare

ASK_M:    MOV     DPTR, #M_BANNER
           CALL    OUT_STRING
           CALL    IN_TIME              ;Get minute & save temp copy.
           MOV     R5, A
           CALL    QUERY
           JNC     ASK_S
           ORL     RTCC, #20h          ;Enable minute compare

ASK_S:    MOV     DPTR, #S_BANNER
           CALL    OUT_STRING
           CALL    IN_TIME              ;Get second & save temp copy.
           MOV     R6, A
           CALL    QUERY
           JNC     ASK_SS
           ORL     RTCC, #40h          ;Enable second compare

ASK_SS:   MOV     DPTR, #SS_BANNER
           CALL    OUT_STRING
           CALL    IN_TIME              ;Get subsecond & save temp copy.
           MOV     R7, A
           CALL    QUERY
           JNC     ASK_X
           ORL     RTCC, #80h          ;Enable subsecond compare.

ASK_X:    MOV     DPTR, #NEW_LINE
           CALL    OUT_STRING

           MOV     RTASS, R7            ;Save new alarm values.
           MOV     RTAS, R6
           MOV     RTAM, R5
           MOV     RTAH, R4

```

```

        ANL    RTCC, #0FDh ;Clear the RTCI flag in case it was
                           ; accidentally set while we were
                           ; manipulating compare bits.
        SETB   ERTCI ;Reenable RTC interrupt.
        LJMP   CHAR_TEST

QUERY:   MOV    DPTR, #COMPARE_Q
        CALL   OUT_STRING
        JNB    RI0, $
        CLR    RI0
        MOV    A, SBUF0
        CALL   OUT_CHAR ;Echo it.
        CJNE   A, #'Y', NO_ENABLE ;If user wants compare, set flag.
        SETB   C
        RET
NO_ENABLE: CLR    C ;User does not want compare, clear flag.
        RET
;*****
;Output routines.
;*****
;This subroutine outputs an ASCII string. The starting point of the string
;is in DPTR, and the terminating character is '0'.
OUT_STRING: PUSH   ACC ;Save accumulator.
CHAR_LOOP: CLR    A ;Clear accumulator for next instruction.
        MOVC   A, @A + DPTR ;Get the next character from the
        JNZ    NXT_CHAR ; string, and if 0, exit.
        POP    ACC ;Restore accumulator.
        RET
NXT_CHAR: CALL   OUT_CHAR ;Next character is valid, so transmit
        INC    DPTR ; it. Increment the data pointer
        JMP    CHAR_LOOP ; to the next position and loop
                           ; back to send character.

;*****
;This subroutine outputs a leading colon for the minute, second, and subsecond
; when displaying the time. When done, it falls through to OUT_DIGIT.
OUT_CDIGIT: MOV    SBUF0, #' ':' ;Display a colon.
        JNB    TI0, $
        CLR    TI0
;This subroutine outputs a hex number in ASCII format through serial port 0.
OUT_DIGIT: MOV    DPTR, #HEX_TABLE
        MOV    R0, A ;Make another copy of value
        SWAP   A ;Do high nibble first
        ANL    A, #0Fh ;Clear unused nibble
        MOVC   A, @A+DPTR ;Get character from table
        CALL   OUT_CHAR ;Transmit the character.

        MOV    A, R0 ;Now do low nibble.
        ANL    A, #0Fh ;Clear unused nibble
        MOVC   A, @A+DPTR ;Get character from table

```

```

        CALL    OUT_CHAR    ;Transmit the character.
        RET                      ;Done

OUT_CHAR:  MOV     SBUF0, A    ;Transmit the character out the serial
          JNB     TI0, $      ; port and wait until complete.
          CLR     TI0
          RET

;*****
;Input routines.
;*****
;IN_TIME takes two decimal characters from the serial port, and formats them
; as a hexadecimal number.
IN_TIME:  CALL    IN_CHAR    ;Get tens digit.
          MOV     B, #0Ah    ;Multiply first digit by 10 and save to
          MUL     AB        ; add to ones digit.
          XCH     A, B
          CALL    IN_CHAR    ;Get ones digit and add it.
          ADD     A, B      ;Acc now has hex value of 2 decimal digit
          RET          ; number. Exit.

IN_CHAR:  JNB     RI0, $      ;Wait for character.
          CLR     RI0
          MOV     A, SBUF0
          CALL    OUT_CHAR    ;Echo character back.
          PUSH    ACC        ;Save copy of A.
          ANL     A, #0F0h   ;If bits 7-4 are not 3h, then character
          CJNE    A, #30h, IN_CHAR ; is not 0-9. Get another character.
          POP     ACC        ;Restore A.
          ANL     A, #0Fh    ;Acc now contains 0-9
          RET

;*****
;RTC_INT - This ISR notifies the user that an alarm has occurred, and gives
; the time of the alarm.
;*****
RTC_INT:  ANL     RTCC, #0FDh ;Clear RTC Interrupt flag.
          MOV     DPTR, #ALARM_MSG ;Display alarm message and time of alarm.
          CALL    OUT_STRING
          CALL    OUT_TIME
          RETI          ;Return

```

## RTC CRYSTAL CONSIDERATIONS

The most important factor in the accuracy of the RTC (or any oscillator) is the characteristics of the oscillator crystal. The DS87C530 is rated for an accuracy of  $\pm 2$  minutes per month over the full operating range of the device. Even higher accuracy can be obtained by controlling the temperature of the device and using the RTC calibration procedures described later. The DS87C530 has been designed to operate with 32.768 KHz RTC crystals with a load capacitance ( $C_L$ ) of 6 pF or 12.5 pF.

Unlike some crystal amplifiers, no external load capacitors are needed with the RTC crystal.

Dallas Semiconductor products are compatible with industry standard crystals. Table 2 shows a number of common 32.768 KHz crystals. This list is by no means exhaustive, and the inclusion or exclusion of any vendor from this list is in no way a comment on the suitability of a specific crystal in a customer's application.

**STANDARD 12.5 PF AND 6 PF RTC CRYSTALS** Table 2

MANUFACTURER	MODEL	$C_L$	PACKAGE
Epson Crystal Corp.	MC-306 32.768K E	6.0 pF	SMT
	MC-306 32.768K A	12.5 pF	SMT
KDS America	DT-26S 32.768 KHz	6 pF	Cylinder
	DT-26S 32.768 KHz	12.5 pF	Cylinder
	DMX-26 32.768 KHz	6 pF	SMT
	DMX-26 32.768 KHz	12.5 pF	SMT
AVX/Kyocera	KF-38G-12P5200	12.5 pF	Cylinder
	KS-309G-12P5200	12.5 pF	SMT

## SELECTING LOAD CAPACITANCE

The value of  $C_L$  has the most bearing on the long-term accuracy of the RTC. This parameter specifies the capacitive load that the crystal needs to "see" across its pins to oscillate at its rated frequency. Note that  $C_L$  is not the capacitance of the crystal itself, but rather the capacitance of the oscillator circuit and any capacitors connected to the crystal. Using a crystal that has a different  $C_L$  than the actual load capacitance of the circuit will affect the frequency of the oscillator. In general, using a crystal with a  $C_L$  that is larger than the load capacitance of the oscillator circuit will cause the oscillator to run faster than the specified nominal frequency of the crystal, and vice versa.

The DS87C530 defaults to a mode which makes it compatible with a 12.5 pF crystal, but can be switched to 6 pF by clearing the RTC Capacitance Select bit X12/6 (TRIM.6). Although both crystal types will remain within the specified accuracy, each has a different advantage. The reduced loading of a 6 pF crystal will reduce the power consumption of the RTC crystal oscillator by 25 to 50 percent, increasing the life of the backup battery. A 12.5 pF crystal, however, is less affected by noise and will maintain a higher accuracy over an extended time. Changing the capacitance of the RTC crystal amplifier has no effect on the system clock crystal attached to the X1 and X2 pins.

## FINE-TUNING THE OSCILLATOR FREQUENCY

Although the DS87C530 RTC is designed to oscillate at exactly 32.768 KHz, variations in the device, crystal, temperature, and board layout can produce minor timing variations. By adjusting the RTC Trim Bits located in the RTC Trim Register (TRIM;96h), the internal capacitance of the RTC circuitry can be slightly adjusted to improve timing accuracy beyond the minimum specified. Although the trim bits do not correspond to an absolute value of capacitance or frequency shift, they provide a relative adjustment.

Please note that under normal circumstances, adjusting the RTC Trim Bits is not necessary. Upon a no-battery reset, the DS87C530 will reset its internal capacitance to a default value which will guarantee the minimum accuracy specified. If you do not require accuracy better than 2 minutes per month, please skip this section.

To aid the user in determining the true frequency of the RTC, a 4096 Hz signal derived from the 32.768 KHz crystal is available on the P1.7 pin by setting the E4K bit (TRIM.7). This can be measured with a frequency counter to determine the RTC frequency. Do not attempt to measure the frequency of the RTC at the leads of the crystal. The capacitance of oscilloscope probes will distort the operation of the crystal and report erroneous val-



ues. The error of the RTC in minutes per month can be calculated from the following formula:

$$(P1.7 \text{ Frequency} - 4096.000 \text{ Hz}) * (10.547) \text{ [minutes/month]}$$

Note that this error is calculated at a specific temperature and voltage. Crystal characteristics change over temperature, and the designer is advised to characterize the error over the system's range of expected operating conditions.

The trim register features extensive protection to avoid accidental corruption. All of the bits of the trim register require a Timed Access procedure to modify them. In addition, writes to the trim register must be done in complementary pairs. Each of the three trim bits has a complement bit which must be set simultaneously. This is to ensure that any writes to the TRIM register are intentional. If an invalid bit sequence is written to the trim bits, the TRIM register will reset to 0x100101 binary. This is the no-battery reset value, except that the X12/6 bit will remain unchanged. The settings of the TRIM bits do not correspond to an absolute value of capacitance or frequency, and are only used to provide a relative adjustment.

To adjust the RTC trim bits, place the device into the target system with the selected crystal and remove any

sources of loading from P1.7. Then attach a frequency counter to P1.7 and perform the following procedure.

1. Perform a Timed Access procedure,
2. Set TRIM.7, E4K, and modify the TRIMx bits, writing their complements to the TRIMx bits in the same instruction. This will enable the external 4096 Hz signal on P1.7,
3. Record the frequency,
4. Repeat steps 1–3 eight times until all combinations of TRM0, TRM1, TRM2 have been measured.

After all the measurements have been taken, the measurement closest to 4096.00 Hz is the most accurate setting of the TRIMx bits. Program this value into the TRIM register for the maximum accuracy. An example program is provided below.

### PROGRAM EXAMPLE: RTC CALIBRATION

The following program example is provided to assist system designers in calibrating their RTC for maximum accuracy. It demonstrates how to set the RTC trim bits and pause the program to allow time to read the frequency output on P1.7.

```
; *****
; Program RTC_CALB.ASM
;
; This program configures the DS87C530 so that the internal RTC frequency can
; be measured. A 4 KHz signal, derived by dividing the 32.768 KHz RTC by 8,
; will be asserted on pin P1.7. The device will step through the 8 settings of
; the RTC trim bits, displaying the current contents of the trim register on
; port 3. A delay of approximately 15 seconds (at 25 MHz) is inserted between
; each setting to allow time to record the frequency.
;
; To calibrate the RTC capacitance, connect a frequency counter to pin P1.7 and
; execute this program. Record the frequency from the counter and the trim bit
; settings as shown on port 3 as it steps through the 8 possible trim settings.
; The setting that produces a frequency closest to 4096 Hz is the most accurate
; setting of RTC capacitance.
; *****

RTCC equ 0F9h ;Real Time Clock Control
TA equ 0C7h ;Timed Access Register
TRIM equ 96h ;RTC Trim Register
P3 equ 0B0h ;Port 3 Latch
```

;These definitions are for 6 pF crystal calibration.

```
TRIM0    equ    95h        ;First trim bit setting    (6 pF)
TRIM1    equ    96h        ;Second trim bit setting   (6 pF)
TRIM2    equ    99h        ;Third trim bit setting    (6 pF)
TRIM3    equ    9Ah        ;Fourth trim bit setting   (6 pF)
TRIM4    equ    0A5h       ;Fifth trim bit setting    (6 pF)
TRIM5    equ    0A6h       ;Sixth trim bit setting    (6 pF)
TRIM6    equ    0A9h       ;Seventh trim bit setting  (6 pF)
TRIM7    equ    0AAh       ;Eighth trim bit setting   (6 pF)
```

;These definitions are for 12.5 pF crystal calibration.

```
;TRIM0    equ    0D5h        ;First trim bit setting    (12.5 pF)
;TRIM1    equ    0D6h        ;Second trim bit setting   (12.5 pF)
;TRIM2    equ    0D9h        ;Third trim bit setting    (12.5 pF)
;TRIM3    equ    0DAh        ;Fourth trim bit setting   (12.5 pF)
;TRIM4    equ    0E5h        ;Fifth trim bit setting    (12.5 pF)
;TRIM5    equ    0E6h        ;Sixth trim bit setting    (12.5 pF)
;TRIM6    equ    0E9h        ;Seventh trim bit setting  (12.5 pF)
;TRIM7    equ    0EAh        ;Eighth trim bit setting   (12.5 pF)
```

```
;
cseg      at    0          ;Reset vector.
          LJMP   START
```

```
cseg      at    100H       ;Start of program
```

```
;
START:    MOV     P3, #0AAh    ;I'm alive message.

          MOV     TA, #0AAh    ;Timed access.
          MOV     TA, #55h
          MOV     RTCC, #01h    ;Start RTC and clear RTC interrupt flag.
```

```
          LCALL   HALFSEC      ;Delay to give RTC oscillator time to
                                ; warm up.
```

; End of initialization. Now step through all the settings of the trim bits.

```
          MOV     R0, #TRIM0    ;Trim setting 0
          LCALL   NEXT_SETTING

          MOV     R0, #TRIM1    ;Trim setting 1
          LCALL   NEXT_SETTING

          MOV     R0, #TRIM2    ;Trim setting 2
          LCALL   NEXT_SETTING

          MOV     R0, #TRIM3    ;Trim setting 3
          LCALL   NEXT_SETTING

          MOV     R0, #TRIM4    ;Trim setting 4
          LCALL   NEXT_SETTING
```

```

MOV      R0, #TRIM5      ;Trim setting 5
LCALL    NEXT_SETTING

MOV      R0, #TRIM6      ;Trim setting 6
LCALL    NEXT_SETTING

MOV      R0, #TRIM7      ;Trim setting 7
LCALL    NEXT_SETTING

MOV      P3, #0FFh       ;Turn on all port 3 pins to signal
DONE:    JMP      DONE    ;we're done.

;*****
;NEXT_SETTING - This subroutine writes the new setting to the RTC trim register,
;               displays the value of the trim register on port 3 for reference,
;               and delays for a period to give time to record the data
;*****
NEXT_SETTING:
MOV      TA, #0AAH       ;Timed access.
MOV      TA, #55h
MOV      TRIM, R0        ;Set E4K and new trim setting.
NOP
MOV      P3, TRIM        ;Output value of trim register.

SEC30:   MOV      R3, #30    ;15 second delay with 25 MHz crystal.
SECLOOP: LCALL    HALFSEC
DJNZ     R3, SECLOOP
RET

;*****
;HALFSEC - This subroutine generates a delay of approximately 0.5 second with
;          a 25 MHz crystal.
;*****
HALFSEC: MOV      R0, #25
OUTER:   MOV      R1, #125
MIDDLE:  MOV      R2, #249
INNER:   NOP
DJNZ     R2, INNER
DJNZ     R1, MIDDLE
DJNZ     R0, OUTER
RET

```

## NOISE AND CRYSTAL LAYOUT GUIDELINES

The crystal inputs of the DS87C530 RTC (RTCX1, RTCX2) have a very high impedance. Unfortunately, this can cause the leads to the crystal to function as antennae, coupling high frequency signals into the RTC circuitry from the rest of the system. This can lead to a distortion of the crystal oscillator signal, resulting in extra or missed clock edges. In most situations high frequency noise will present the greatest problem, causing the clock to run fast.

The following procedure can be used to determine if noise is the cause of the inaccuracy of a RTC:

1. Power the system up and synchronize the RTC to a known, accurate clock,
2. Remove  $V_{CC}$  to the device (but maintain  $V_{bat}$ ),
3. Wait for a long period of time (24 hours),
4. Apply  $V_{CC}$ , read the RTC, and compare to the known, accurate clock,
5. Resynchronize the RTC to the known, accurate clock,
6. Keep system powered up and wait for the same period of time in step 3,
7. Read RTC and compare to the known, accurate clock.

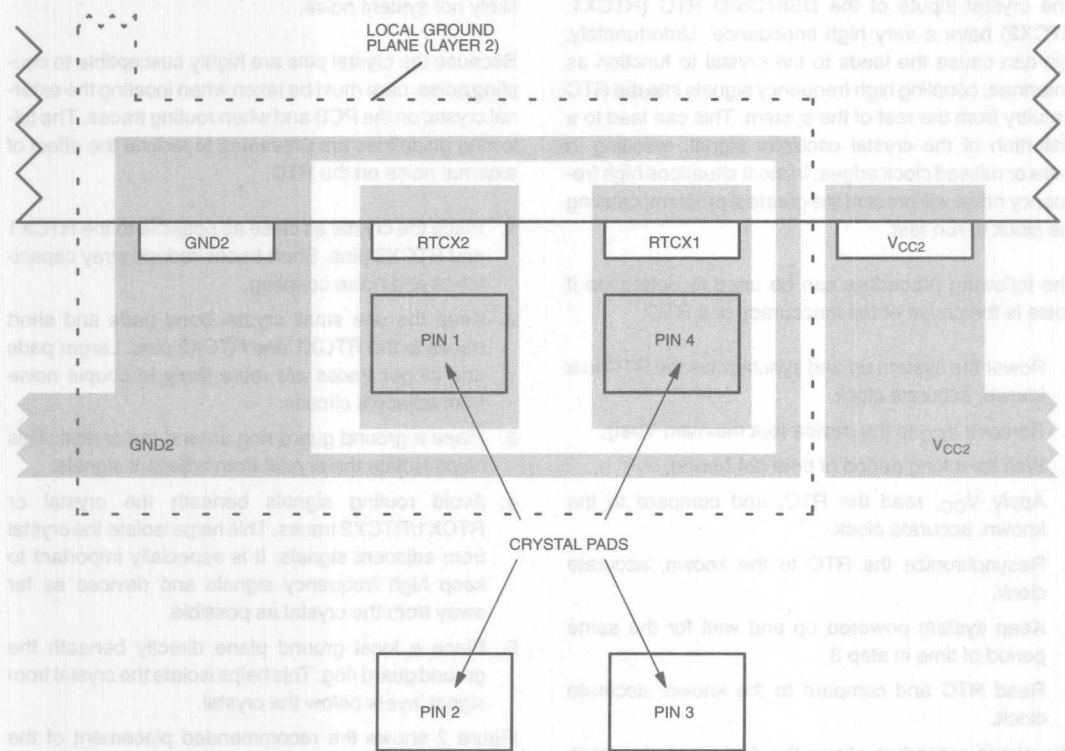
The above procedure allows the designer to measure the inaccuracy of the clock both when the system is operating and when it is powered down. If the clock appears less accurate when powered up, the most likely culprit is system noise. If the inaccuracy remains

whether the system is on or off, then the cause is most likely not system noise.

Because the crystal pins are highly susceptible to coupling noise, care must be taken when locating the external crystal on the PCB and when routing traces. The following guidelines are presented to reduce the effect of external noise on the RTC.

1. Place the crystal as close as possible to the RTCX1 and RTCX2 pins. Short traces reduce stray capacitance and noise coupling.
2. Keep the use small crystal bond pads and short traces to the RTCX1 and RTCX2 pins. Larger pads and longer traces are more likely to couple noise from adjacent circuits.
3. Place a ground guard ring around the crystal. This helps isolate the crystal from adjacent signals.
4. Avoid routing signals beneath the crystal or RTCX1/RTCX2 traces. This helps isolate the crystal from adjacent signals. It is especially important to keep high frequency signals and devices as far away from the crystal as possible.
5. Place a local ground plane directly beneath the ground guard ring. This helps isolate the crystal from signal layers below the crystal.

Figure 2 shows the recommended placement of the RTC crystal, guard ring, and ground plane. The illustration shows one common orientation for a 4-pin surface mount crystal, but pin orientations will vary between manufacturers and package types.

**EXAMPLE CRYSTAL PLACEMENT ON PCB Figure 2**



# DALLAS SEMICONDUCTOR

## Application Note 80 Using the High-Speed Micro's Watchdog Timer

### INTRODUCTION

Today, microcontrollers are being used in harsh environments where electrical noise and electromagnetic-magnetic interference (EMI) are abundant. In environments like this, it is beneficial if the system contains resources to help ensure proper operation. In many systems, a commonly used technique for verifying proper operation is the incorporation of a watchdog timer.

A watchdog timer is fundamentally a time measuring device that is used in conjunction with, or as part of, a microprocessor and is capable of causing the microprocessor to be reset. In a properly designed system, the watchdog will cause a reset when the microprocessor is not operating correctly, thereby eliminating the faulty condition. In a typical application, the watchdog timer is configured to reset the processor after a predetermined time interval. If the processor is operating correctly, it will restart the watchdog before the end of the interval. After being restarted, the watchdog will begin timing another predetermined interval. If the watchdog is not restarted by the processor before the end of the interval, a watchdog time-out occurs. This results in the processor being reset. If the system software has been designed correctly, and there has been no hardware failure, the reset will cause the system to operate properly again. Of course, the reset condition must be a safe state. For instance, it would not be wise to have the reset state of a disk drive controller enabling the write head.

Many systems have been designed using an external watchdog timer. The need for this additional external component is eliminated, however, with the DS80C320. The DS80C320 contains its own very capable, internal, watchdog timer. The features and the use of this watchdog timer are the subject of this application note.

### GENERAL USE OF A WATCHDOG TIMER

The primary application of a watchdog timer is as a system monitor (as discussed in detail in the section below). With a watchdog timer, a system can be

designed to be very good at detecting and correcting an out of control microprocessor. A system using a watchdog timer is particularly well suited to detecting bit errors. Momentary bit errors can be caused by such things as soft memory failures and electromagnetic discharges into memory devices and their interfaces. These can cause temporary bit polarity flipping of data into and out of the processor. When this occurs while fetching program information, the microprocessor will begin executing erroneous code. Potentially, the processor could begin executing operands instead of op-codes. When the processor begins executing this bad code, it will not properly execute the code that restarts the watchdog. After the time-out interval, the watchdog will cause a processor reset. In a properly designed system, the reset will correct the error.

Regardless of how capable a watchdog timer might be, it cannot resolve all reliability issues. There are certain failures that cannot be corrected by a reset. For instance, a watchdog timer cannot prevent the corruption of data. In its basic form, the watchdog restart is dependent on proper program execution, and generally, does not depend on the values in data memory. Unless corruption of data affects program flow, or some extra measures are taken, data corruption will not cause a watchdog time-out. Of course, self-diagnostic software can be written in such a way as to make restarting the watchdog contingent on verification of data memory. While this approach can be very effective and is quite common, it is beyond the scope of this document to discuss in detail.

Also note that a watchdog timer cannot detect a fault instantaneously. By definition, the watchdog timer must reach the end of a predetermined time interval before it resets the processor. This fact explains why a minimum possible time-out interval should be selected. In this way, a minimum amount of time expires before an out of control condition is corrected.

## THE WATCHDOG AS A SYSTEM SUPERVISOR

The most common use of the High Speed Micros watchdog timer is as a system supervisor. While it can be used in a number of different ways, some of which will be discussed in this document, system supervisor is the most common application. In system supervisor mode, the timer is restarted periodically by the processor as described above. If the processor runs out of control, the watchdog will not be restarted, it will time-out, and subsequently will cause the processor to be reset.

In the High Speed Micro, the watchdog timer is driven by the main system clock that is supplied to a series of dividers. The divider output is selectable, and determines the interval between time-outs. When the time-out is reached, an interrupt flag will be set, and if enabled, a reset will occur 512 clocks later. The interrupt flag will cause an interrupt to occur if its individual enable bit is set and the global interrupt enable is set. The reset and interrupt are completely discrete functions that may be acknowledged or ignored, together or separately for various applications.

When using the watchdog timer as a system monitor, the watchdog's reset function should be used. If the interrupt function were used, the purpose of the watchdog would be defeated. To explain, assume the system is executing errant code prior to the watchdog interrupt.

The interrupt would temporarily force the system back into control by vectoring the CPU to the interrupt service routine. Restarting the watchdog and exiting by an RETI or RET would return the processor to the lost position prior to the interrupt. By using the watchdog reset function, the processor is restarted from the beginning of the program, and thereby placed into a known state.

This is not to say that the DS80C320 watchdog's interrupt function is not useful for the system monitor application. Since the reset occurs 512 clocks after the interrupt, a short interrupt service routine can be used to store critical variables before the reset occurs. This may allow the system to return to proper operation in a state that more closely resembles the conditions before the failure. Of course, if the data is the source of the error, storing it without correction would be of no benefit. For any specific system, the approach taken is a function of the system and the level of reliability required.

As mentioned above, the watchdog timer in the DS80C320 is driven by the main system clock that is passed through a series of dividers. The divider output may be selected by the user, allowing a time-out of  $2^{17}$ ,  $2^{20}$ ,  $2^{23}$ , or  $2^{26}$  clocks. If enabled, a reset of the processor will occur 512 clocks later. Table 1 shows the reset time intervals associated with different crystal frequencies.

**DS80C320 WATCHDOG RESET TIME INTERVALS** Table 1

CLOCKS	@1.832 MHz	@11.059 MHz	@12 MHz	@25 MHz
$2^{17} + 512$	71.83 ms	11.90 ms	10.97 ms	5.26 ms
$2^{20} + 512$	572.6 ms	94.86 ms	87.42 ms	41.96 ms
$2^{23} + 512$	4.58 s	758.6 ms	699.1 ms	335.6 ms
$2^{26} + 512$	36.63 s	6.07 s	5.59 s	2.68 s

As can be seen, there is a range of time-out intervals available. The interval selected should be based on several issues. The first objective is to select an interval that represents the maximum time the processor can be allowed to run out of control. For example, a system that issues a position command to a robotic arm every 500 milliseconds, ideally, would not use a time-out interval greater than this. Keeping the time-out interval shorter, ensures that there will be at most one bad command issued to the arm.

The other primary concern in setting the watchdog time-out interval is the ability to locate the restart commands within the system software. This can be a very complicated issue depending on the nature of the system software. The most desirable approach is to have a single location within a single main loop of the system software that restarts the watchdog timer. The time required to pass through the main program loop will determine the required time-out interval.

The above approach assumes that the system software flow is linear enough to allow it. Some programs are too convoluted and their flow is too non-linear to allow this approach. With a program structure like that, it is difficult to locate the correct points for watchdog restarts. One possible solution to this problem is to use the DS80C320's watchdog timer itself to assist in determining the appropriate restart locations. This method uses the watchdogs interrupt capability, and is described in detail in a section below.

In some systems, the software is too complex or the program flow is too variable to allow a complete and thorough analysis. It may be impossible to determine that all program paths have been covered by a watchdog restart. In this case, a different approach may be used. In this case, diagnostic software may be developed to test the system. This diagnostic software will be called at periodic intervals, perhaps using the interrupt feature of the watchdog timer. If the diagnostics pass, the watchdog is restarted. If not, the watchdog times out and the processor is reset. Of course in this case, the test must be thorough enough to be effective. The exact approach used in a given system may be any of the above or some combination of each, as appropriate for the application.

## WATCHDOG RESET EXAMPLE

A short program illustrating most of the basic watchdog timer functions is shown below. This program illustrates how to initialize the watchdog timer so that when it times out, it will cause a reset.

The program illustrates one of the DS80C320 watchdog timer's unique features. Software that changes the watchdog's operation must perform a timed access operation. A timed access operation is a sequence of steps that must be executed together, in sequence, otherwise the access fails. The example program shows the timed access being used for restarting the watchdog and enabling its reset. As can be seen, the value 0AAh is first written to the timed access register (TA). Next, the value 055h is written to the TA register. Finally, the protected bit is modified. These instructions must be executed in the sequence shown with no interruption to gain access to the protected bit. Further details on timed access operation may be found in the High Speed Micro Users Guide. The watchdog timer bits that are protected by the timed access procedure are the Enable Watchdog Timer Reset (EWT = WDCON.1) bit, the WatchDog Interrupt Flag (WDIF = WDCON.3) bit, and the Restart Watchdog Timer (RWT = WDCON.0) bit.

```
; *****
;                               WD_RST.ASM Program
;
;
;   This program demonstrates the use of the watchdog timer in
;   the DS80C320. It uses the timer's reset capability. When
;   running, the program sets port 1's pins low to indicate
;   the processor is idle waiting for the watchdog to time-out. When
;   the watchdog times out, the processor is reset causing the port
;   pins to return high. A delay is written into the program so that
;   the port pins will be high long enough to be seen if attached to
;   LEDs.
;
; *****
;   Reset Vector
;
;   ORG          00h
;   SJMP         START
;
; *****
;
;   Main program body
;
;   ORG          080h
;
; *****
```



```

;      each time the watchdog's Interrupt Service Routine is entered.
;
;
;
$MODS320
;
; *****
;
;      Reset Vector
;
;      ORG      00h
;      SJMP     START
;
; *****
;
;      Watchdog Interrupt Vector
;
;      ORG      063h
;
;      MOV      TA, #0AAh ; Restart watchdog timer
;      MOV      TA, #055h ; using timed
;      SETB     RWT       ; access.
;
;      MOV      TA, #0AAh ; Clear watchdog interrupt flag
;      MOV      TA, #055h ; using timed
;      CLR      WDIF      ; access.
;
;      CPL      A          ; Complement port 1 to show the
;      MOV      P1, A      ; interrupt routine was entered.
;
;      RETI              ; Return from interrupt.
;
; *****
;
;      Main program body
;
;
;      ORG      080h
;
START:
;      ORL      CKCON, #040h; Set Watchdog time-out period 2**20
;                               ; (approximately 94.8 mS @ 11.059 MHz)
;
;      MOV      TA, #0AAh ; Restart Watchdog timer
;      MOV      TA, #055h ; using timed
;      SETB     RWT       ; access.
;
;      SETB     EWDI      ; Enable Watchdog Interrupt and
;      SETB     EA        ; set global interrupt enable
;
;
Here:  MOV      PCON, #01 ; Go to Idle mode and wait
      SJMP     Here     ; After interrupt, go back to idle

```



```

;
;
; *****
;

```

END

## THE WATCHDOG TIMER AS AN AID IN LOCATING RESTART INSTRUCTIONS

As discussed above, locating the watchdog restart instructions in the system software can sometimes be difficult. The structure of the system software and the complexity of its flow, determines the task's level of difficulty. In the DS80C320, the watchdog timer itself can be used to assist in this activity. The general approach for this is to allow the watchdog to cause an interrupt, and from within the service routine, determine where in the code the interrupt occurred. By placing the watchdog restart instructions prior to this point, you can be assured that the watchdog will be restarted before the time-out (when the software flow follows this particular branch). This process is repeated until no more watchdog interrupts occur. If the program flow is linear and not data dependent, the system will function as desired.

The previous software example provides most of the software necessary to perform this function. As a first step however, the desired maximum time-out interval should be determined, and the code modified for this value. As always, the time-out selected is a function of the system and how long the micro can be allowed to run out of control. After modifying the software to initialize the desired watchdog time-out interval, the following instructions should be added to the Interrupt Service Routine. They will cause the processor to display the address of the instruction that would have been executed if the interrupt had not occurred. If this display mechanism is not convenient for the system implementation, the address can be converted to ASCII and output on one of the serial ports.

```

MOV  R0, SP      ; Get SP contents
MOV  P3, @R0     ; Display high address byte
DEC  R0          ; Point to low address byte
MOV  P1, @R0     ; Display low address byte
SJMP $          ; Stop here

```

The instructions above, move the contents of the Stack Pointer to R0 that is then used to point to the data

pushed onto the stack when the interrupt was acknowledged. This address reflects the next instruction that would have been executed if the interrupt had not occurred. The high byte of the address is displayed on Port 3 pins, and the low byte of the address is displayed on Port 1 pins. If instructions to restart the watchdog timer are placed before this address, the watchdog will never reach time-out.

## SUMMARY

When designing a system using a watchdog as a monitor, there are a number of considerations that must go into an effective design. First, the maximum time that the processor can run out of control will determine the maximum watchdog time-out period. Once the time-out period is determined, the system software must be analyzed to determine where to locate the watchdog restart instructions. For an effective design, the number of watchdog restarts should be kept to a minimum, and some consideration should be given to the likelihood of incorrectly executing a restart. As mentioned previously, some system software is too convoluted or data dependent to ensure that all software flow paths are covered by a watchdog restart. This may dictate that a self-diagnostic software approach might be required. If there is an expected failure mechanism such as a periodic EMI burst or power supply glitch, the watchdog time-out should consider this period.

For the watchdog reset to be an effective error correction mechanism, the reset state of the processor must be safe. In some applications, the watchdog's interrupt capability might be used to manipulate data or the stack before the reset to ensure that the processor will function properly after the reset.

By carefully considering the above aspects, a system can be designed using a watchdog timer that will operate in very harsh environments.

# DALLAS SEMICONDUCTOR

## Application Note 81 Memory Expansion with the High-Speed Microcontroller Family

### OVERVIEW

All members of the High-Speed Microcontroller Family are designed to directly address up to 64KB of program and data memory. Occasionally, however, an application will require more memory than is either available on-chip or through the use of the 64KB memory map. The High-Speed Microcontroller Family includes many features which make it easy to address program and/or data memory greater than 64KB. Bit-addressable I/O ports allow single instruction modification of control lines, which can be used to bank switch or page between multiple memory devices. The ROMSIZE™ feature allows easy memory resizing for devices with on-chip memory.

This application note discusses the expansion of both program and data memory. It is subdivided into three main categories: expanding program memory of ROM-less devices beyond 64KB, using the ROMSIZE™ feature to expand on-chip program memory up to and beyond 64KB, and expanding data memory. It begins with an introduction to bank switching and software support techniques.

### BANK SWITCHING THEORY

Expanded memory access beyond 64KB is most frequently done through bank switching. This technique uses one or more general purpose I/O lines as decode lines to address more memory. If a single large-capacity memory device is used, the additional signals can be used directly as address lines. If several smaller capacity memory devices are used, the signals can be used as chip selects. The basic unit of memory switched by the decode logic is called a bank or page. For example, if an I/O line was used to switch between two 64KB EPROMs, the memory would consist of two 64KB pages.

Probably the biggest obstacle in using a paged memory scheme is the placement of the interrupt vector table. During most of the device operation, software can perform an orderly switch between pages. When an interrupt occurs, however, the device will immediately jump to the appropriate vector address, below 0070h. The software has no control over the bank configuration at this point, and the device will attempt to jump to the low end of the current bank to seek the vector table.

There are two approaches to solving this problem. The simplest is to duplicate the interrupt vector table at the low end of each page. In this way, the interrupt vector table will be available at all times, regardless of the current memory configuration. There are a number of disadvantages to this approach, however. It is an inefficient use of program memory as the interrupt vector table (approximately 120 bytes) and often the interrupt service routines, must be duplicated on every page. Also, some compilers do not directly support duplication of data across pages, complicating program generation. A more efficient approach is to reserve some lower portion of memory, which includes the interrupt vector table, so that it is not paged. This "common area" is directly accessible from any expanded bank without modification of the bank selection mechanism. Any time the processor performs a code fetch at this common area, hardware forces the memory to access this area, regardless of the current page. Careful design will allow the previous bank address to be saved, so that the device will return automatically when the operation in the common memory is complete. Interrupt service routine execution times can be reduced by locating them along with the interrupt vector table in the common area located at the low end of memory. Most of the examples in this application note designate the lower region of memory as the common area.

## SOFTWARE SUPPORT FOR MEMORY EXPANSION

For a paged memory scheme to work, it is necessary to fragment the code into pages and provide a means for the software to switch between pages. Care must be exercised when switching pages that the instruction flow will not be disturbed. There are two main approaches to this. The first is to switch pages "on the fly." This is demonstrated in the simple page expansion example which follows. This approach causes program execution to jump directly from one expanded page to another expanded page. One must exercise caution so that the bank switch will occur at a location that corresponds to the start of the next instruction in the next bank. Failure to correctly align the instructions may cause the next opcode fetch to occur in the middle of a multi-byte instruction, resulting in complete loss of program control.

A better approach is to change banks from a location that will be unaffected by the change. This is most often a common or unpaged location in memory, such as a reserved location where the interrupt vector table is located. Any access to lower memory is automatically switched into the common memory by hardware. This eliminates the code alignment difficulties with the simple page expansion above, and the need to duplicate interrupt vectors and/or interrupt service routines on each memory page.

Many compilers and linkers directly support bank switching, and many of them include library functions for page switching. The documentation accompanying your compiler will provide information concerning its expanded memory support. Brief examples of assembly language support are listed after some of the examples below.

Care must be exercised if multiple pages are programmed into a single EPROM. Many EPROM programmers calculate program offsets using the address specified in the file, which can lead to misplaced code. For example, suppose that a paging scheme involves

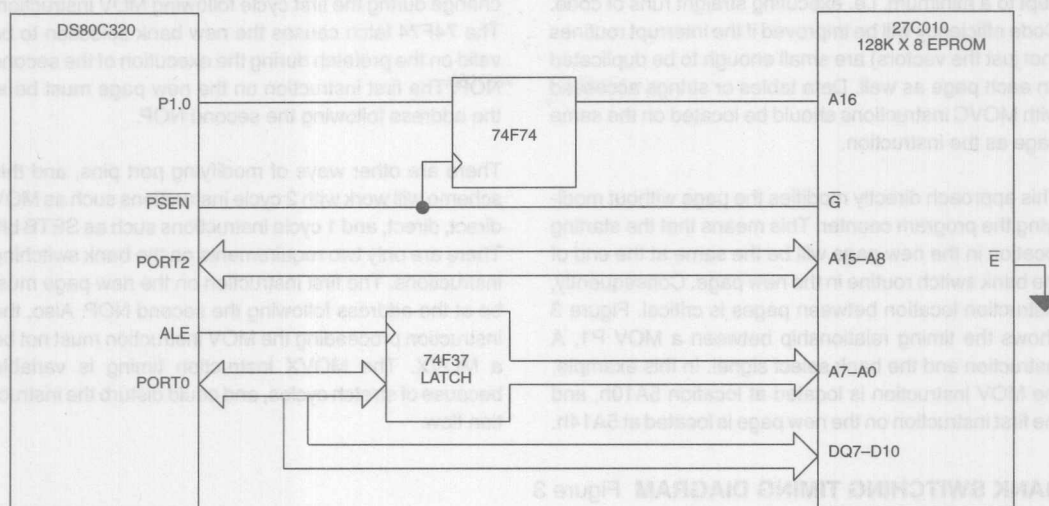
pages of code mapped into program space from 8000h to FFFFh. If the designer wished to locate a page at 10000h in the EPROM, he or she would normally select an offset of 10000h in the EPROM when loading the file into the programmer. All addresses in the hex file begin at 8000h, however, which the device programmer would add to 10000h. This would inadvertently place the page at 18000h, which was not the intended result. Various device programmers implement offsets in different ways, and the designer is advised to consult the documentation accompanying the device programmer for the best solution.

## ROMLESS PROGRAM EXPANSION

The absence of on-chip program memory makes expanding the program memory of the DS80C320 relatively simple. Three approaches to expanding program memory are presented here. The first involves expansion of relatively small amounts of memory by duplicating vector tables and overlapping pages. The second example uses a common bank for interrupt vectors and interrupt service routines, and pages memory using several general purpose I/O lines. The last example uses a latched address to address large amounts of memory without using additional general purpose I/O lines.

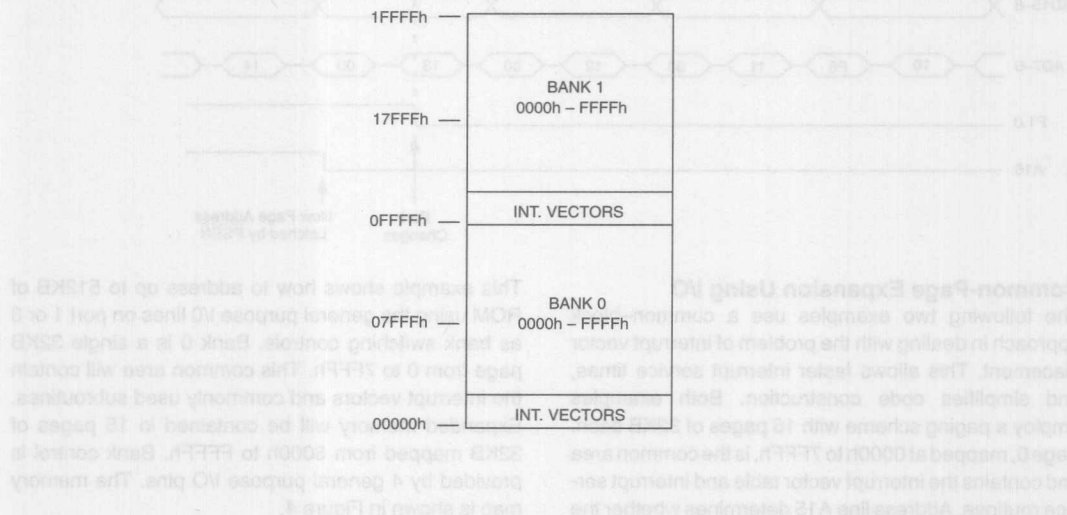
### Simple Page Expansion

This example shows the simplest way of adding relatively small amounts of program memory. A single general purpose I/O line is used to provide up to 128K bytes of program memory. A single 27C010 128K byte EPROM is used, and is divided into two overlapping memory blocks. One general purpose I/O line, P1.0 in this case, is used to provide bank switch control. It is latched by a 74F74, which is clocked on the rising edge of the  $\overline{\text{PSEN}}$  signal. This synchronizes the bank switch with the memory cycle. This approach can be extrapolated to add even greater amounts of memory by using additional I/O lines. The hardware configuration for this example is shown in Figure 1.

**SIMPLE PAGE EXPANSION EXAMPLE HARDWARE** Figure 1

The simplicity of the hardware comes at the cost of some software complexity, however. This example uses two banks, both of which contain the interrupt vector tables in the lower portion of memory. This is necessary because when the device is reset, P1.0 will be high, forcing the reset vector address to 10000h. Also, an interrupt may occur while executing code from either

page, so the interrupt vectors must be available without software intervention. The interrupt vector table consumes approximately 115 bytes from locations 00000h to 00070h, and 10000h to 10070h. Additional space may be required if duplication of interrupt service routines is desired on each page.

**SIMPLE PAGE EXPANSION EXAMPLE MEMORY MAP** Figure 2

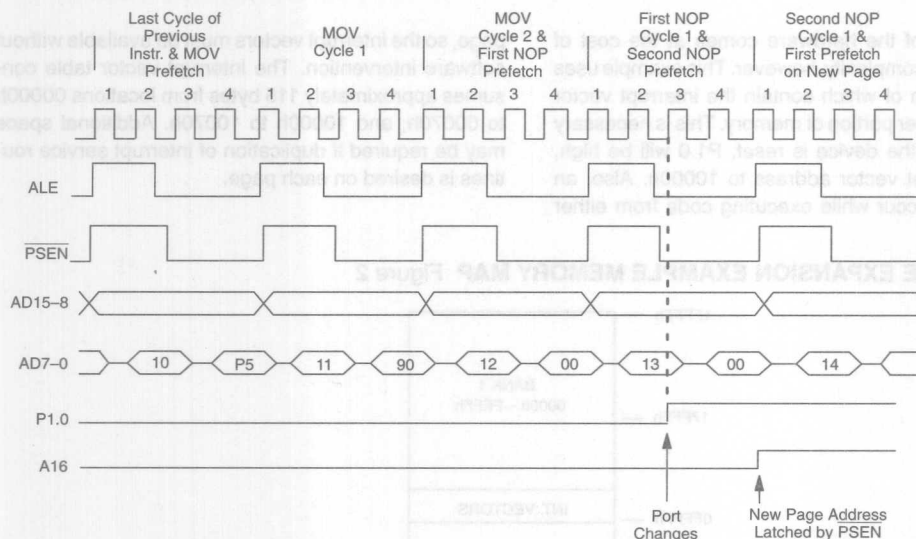
This approach is most effective when page switching is kept to a minimum, i.e. executing straight runs of code. Code efficiency will be improved if the interrupt routines (not just the vectors) are small enough to be duplicated on each page as well. Data tables or strings accessed with MOVC instructions should be located on the same page as the instruction.

This approach directly modifies the page without modifying the program counter. This means that the starting location in the new page will be the same at the end of the bank switch routine in the new page. Consequently, instruction location between pages is critical. Figure 3 shows the timing relationship between a MOV P1, A instruction and the bank select signal. In this example, the MOV instruction is located at location 5A10h, and the first instruction on the new page is located at 5A14h.

The port pin which controls the bank selection will change during the first cycle following MOV instruction. The 74F74 latch causes the new bank selection to be valid on the prefetch during the execution of the second NOP. The first instruction on the new page must be at the address following the second NOP.

There are other ways of modifying port pins, and this scheme will work with 2 cycle instructions such as MOV direct, direct, and 1 cycle instructions such as SETB bit. There are only two requirements on the bank switching instructions. The first instruction on the new page must be at the address following the second NOP. Also, the instruction proceeding the MOV instruction must not be a MOVX. The MOVX instruction timing is variable because of stretch cycles, and could disturb the instruction flow.

**BANK SWITCHING TIMING DIAGRAM** Figure 3



### Common-Page Expansion Using I/O

The following two examples use a common-block approach in dealing with the problem of interrupt vector placement. This allows faster interrupt service times, and simplifies code construction. Both examples employ a paging scheme with 16 pages of 32KB each. Page 0, mapped at 0000h to 7FFFh, is the common area and contains the interrupt vector table and interrupt service routines. Address line A15 determines whether the common block, or one of the 15 expanded pages is addressed.

This example shows how to address up to 512KB of ROM using the general purpose I/O lines on port 1 or 3 as bank switching controls. Bank 0 is a single 32KB page from 0 to 7FFFh. This common area will contain the interrupt vectors and commonly used subroutines. Expanded memory will be contained in 15 pages of 32KB mapped from 8000h to FFFFh. Bank control is provided by 4 general purpose I/O pins. The memory map is shown in Figure 4.



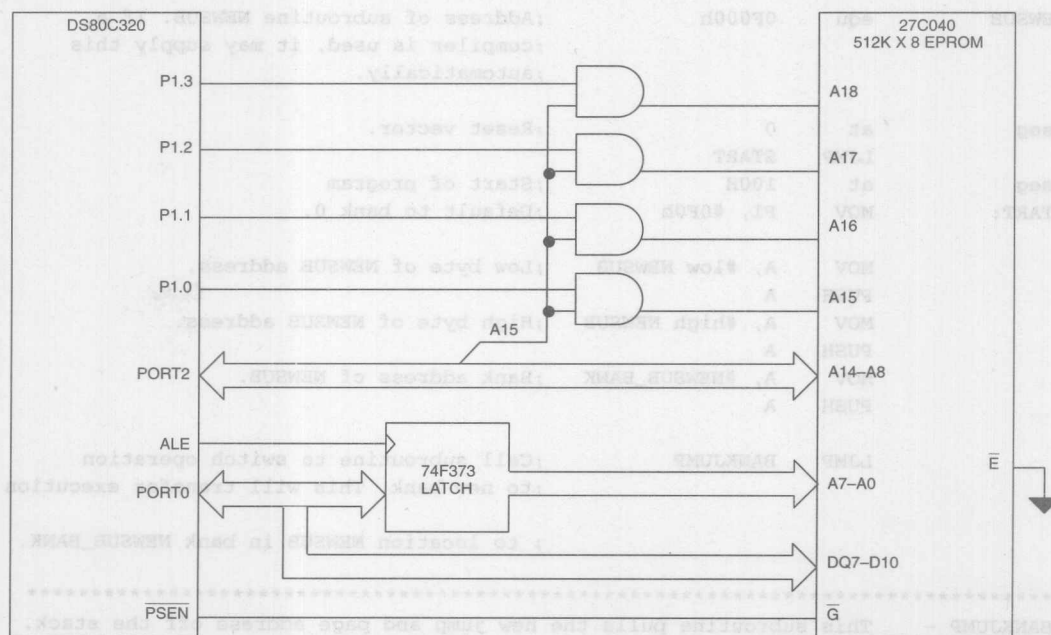
## COMMON-PAGE EXPANSION MEMORY EXAMPLE MEMORY MAP Figure 4

EPROM Address	Code Address and P1.3-0	EPROM Address	Code Address and P1.3-0	EPROM Address	Code Address and P1.3-0	EPROM Address	Code Address and P1.3-0
1FFFFh	BANK 3 8000h – FFFFh P1=x3h	3FFFFh	BANK 7 8000h – FFFFh P1=x7h	5FFFFh	BANK 11 8000h – FFFFh P1=xBh	7FFFFh	BANK 15 8000h – FFFFh P1=xFh
17FFFh	BANK 2 8000h – FFFFh P1=x2h	37FFFh	BANK 6 8000h – FFFFh P1=x26h	57FFFh	BANK 10 8000h – FFFFh P1=xAh	77FFFh	BANK 14 8000h – FFFFh P1=xEh
0FFFFh	BANK 1 8000h – FFFFh P1=x1h	2FFFFh	BANK 5 8000h – FFFFh P1=x5h	4FFFFh	BANK 9 8000h – FFFFh P1=x9h	6FFFFh	BANK 13 8000h – FFFFh P1=xDh
07FFFh	BANK 0 0000h – 7FFFh P1=xxh	27FFFh	BANK 4 8000h – FFFFh P1=x4h	47FFFh	BANK 8 8000h – FFFFh P1=x8h	67FFFh	BANK 12 8000h – FFFFh P1=xCh
00000h		20000h		40000h		60000h	

The hardware configuration is shown in Figure 5. Bank control is provided by P1.0–3, decoded by 4 AND gates, requiring only a single IC package. When A15 is low, the device is forced to access only the lower 32KB of

memory. This removes the need for software intervention when accessing the interrupt vector table in low memory. This example uses an 27C040 512KB EPROM.

## DS80C320 EXPANDED MEMORY EXAMPLE HARDWARE CONFIGURATION Figure 5



The following software example shows an assembly language routine to jump to a new location in any bank using I/O lines as shown in Figure 5. Before calling the bank switch subroutine, software pushes the new address and bank number onto the stack. It then calls a subroutine that pops the new bank address from the

stack and places it on P1.0–3. The stack is then modified so that the subsequent RET instruction will return to the new program location. This is a simple demonstration of one of many possible ways to effect a bank-switch in assembly code.

#### PROGRAM EXAMPLE: JUMPING BETWEEN BANKS USING I/O

```

;*****
;Program BANKJMP1.ASM
;
;This program demonstrates one possible way to jump between banks in a common-
;page memory expansion configuration using I/O. Four general purpose I/O pins
;at P1.0-3 are used to control external bank selection.
;
;Software pushes the address in the new bank, as well as the new bank address
;onto the stack. The subroutine BANKJUMP, located at a non-paged address,
;modifies the page selection, and then repositions the stack pointer to the
;new address which was previously pushed onto the stack. The RET is then
;executed, resuming operation from the new address in the new page. Note that
;BANKJUMP can be called from any area in program memory.
;*****
;Equate table
BANKMASK      equ      0F0h          ;P1.3-0 are used for bank selection.
NEWSUB_BANK    equ      03h          ;Subroutine NEWSUB is located on page 03h.

NEWSUB        equ      0F000h        ;Address of subroutine NEWSUB. If a
;                                     ;compiler is used, it may supply this
;                                     ;automatically.

cseg          at          0           ;Reset vector.
              LJMP        START

cseg          at          100H        ;Start of program
START:        MOV         P1, #0F0h   ;Default to bank 0.

              MOV         A, #low NEWSUB ;Low byte of NEWSUB address.
              PUSH        A
              MOV         A, #high NEWSUB ;High byte of NEWSUB address.
              PUSH        A
              MOV         A, #NEWSUB_BANK ;Bank address of NEWSUB.
              PUSH        A

              LJMP        BANKJUMP     ;Call subroutine to switch operation
;                                     ;to new bank. This will transfer execution
;                                     ;to location NEWSUB in bank NEWSUB_BANK.

;*****
;BANKJUMP - This subroutine pulls the new jump and page address off the stack.
;           ;It modifies P1.0-3, and then modifies the stack pointer to point
;           ;to the new address. It then RETURNS to the new jump location.

```

```

; This subroutine must be placed in a common, unpagged memory area.
;*****
BANKJUMP: CLR      EA                ;Disable all interrupts.
          POP      A                ;Get the new page address off the stack
          ANL      P1, #BANKMASK    ;and modify only P1.0-3 to new bank
          ORL      P1, A            ;address.
          SETB     EA                ;Reenable interrupts.
          RET                     ;Stack pointer is now at new return
                                   ;address. Program will begin executing
                                   ;from new bank.

```

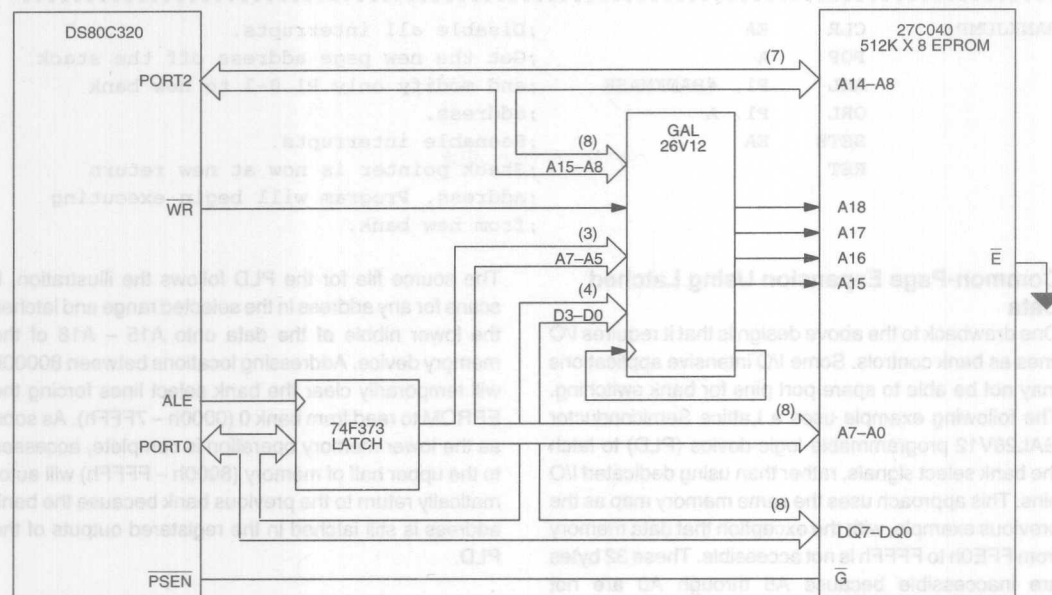
### Common-Page Expansion Using Latched Data

One drawback to the above design is that it requires I/O lines as bank controls. Some I/O intensive applications may not be able to spare port pins for bank switching. The following example uses a Lattice Semiconductor GAL26V12 programmable logic device (PLD) to latch the bank select signals, rather than using dedicated I/O pins. This approach uses the same memory map as the previous example, with the exception that data memory from FFE0h to FFFFh is not accessible. These 32 bytes are inaccessible because A5 through A0 are not decoded, allowing a smaller, lower cost PLD to be used. Decoding more address lines would reduce the amount of inaccessible data memory, but would require a more complicated decoding mechanism.

The GAL26V12 performs the bank switching function based on a write to MOVX data memory. Any write to data memory from FFE0h to FFFFh is decoded, and the lower four bits of the data written to that address are used to configure the bank switch select lines. The hardware configuration is shown in Figure 6.

The source file for the PLD follows the illustration. It scans for any address in the selected range and latches the lower nibble of the data onto A15 – A18 of the memory device. Addressing locations between 80000h will temporarily clear the bank select lines forcing the EPROM to read from bank 0 (0000h – 7FFFh). As soon as the lower memory operation is complete, accesses to the upper half of memory (8000h – FFFFh) will automatically return to the previous bank because the bank address is still latched in the registered outputs of the PLD.

Although a variety of PLDs are suitable for this application, any device used must reset its outputs to 0 on power-up. This is necessary because upon power-up the device must be able to access the reset vector located at 0000h in bank 0. When selecting a PLD, the designer should be aware that many standard programmable logic devices are designed so that their outputs go high upon reset.

**DS80C320 LATCHED ADDRESS MEMORY HARDWARE EXAMPLE Figure 6**

The following software example shows an assembly language routine to jump to a new location in any bank using latched data as shown in Figure 6. Before calling the bank switch subroutine, software pushes the new address and bank number onto the stack. It then calls a

subroutine that pops the new bank address from the stack and writes it out to location FFFFh, where it is latched as the new bank address. The stack is then modified so that the subsequent RET instruction will return to the new program location.

#### PROGRAM EXAMPLE: JUMPING BETWEEN BANKS USING LATCHED ADDRESSING

```

;*****
;Program BANKJMP2.ASM
;
;This program demonstrates one possible way to jump between banks in a common
;page memory expansion configuration using latched addressing.
;
;Software pushes the address in the new bank, as well as the new bank address
;onto the stack. The subroutine BANKJUMP, located at a non-paged address,
;modifies the page selection, and then repositions the stack pointer to the
;new address which was previously pushed onto the stack. The RET is then
;executed, resuming operation from the new address in the new page. Note that
;BANKJUMP can be called from any area in program memory.
;*****
;Equate table
LATCH_ADR    equ    0FFFFh    ;Address of bank select latch.
NEWSUB_BANK  equ    03h       ;Subroutine NEWSUB is located on page 03h.
NEWSUB       equ    0F000h     ;Address of subroutine NEWSUB. If a compiler is
;                               ; used, it may supply this automatically.

```

```

cseg      at      0      ;Reset vector.
          LJMP     START

cseg      at      100H    ;Start of program

START:    MOV      A, #low NEWSUB ;Low byte of NEWSUB address.
          PUSH     A
          MOV      A, #high NEWSUB ;High byte of NEWSUB address.
          PUSH     A
          MOV      A, #NEWSUB_BANK ;Bank address of NEWSUB.
          PUSH     A

          LJMP     BANKJUMP      ;Call subroutine to switch operation
                                ; to new bank.

;*****
;BANKJUMP - This subroutine pulls the new jump and page address off the stack.
;          It writes the new page address out to the PLD, where it is latched
;          as the new page address. The RET then causes execution to the address
;          specified in the function that called this routine. This subroutine
;          must be placed in a common, unpaged memory area.
;*****
BANKJUMP: CLR      EA      ;Disable all interrupts.
          ; point to page address.
          POP      A      ;Get the new page address off the stack.
          MOV      DPTR, #LATCH_ADR ;Write page address out to latch.
          MOVX     @DPTR, A.
          SETB     EA      ;Restore interrupts.
          RET        ;Stack pointer is now at new return
                    ; address. Program will begin executing
                    ; from new bank.

```

The source file for the GAL26V12 PLD is presented to aid the designer in developing his or her own devices.

The file was written in the CUPL language, but can be easily be modified to work with other assemblers.

### PROGRAM EXAMPLE: GAL PROGRAM FILE

```

Name MEM_EXP;
Device g26v12;

/*****
/** This CUPL file will program a GAL26V16 as a bank switching    **/
/** latch to address up to 512 kbytes of program memory. Any     **/
/** MOVX write to addresses FFE0h through FFFFh will latch D0 - D3 **/
/** as the new bank selection.                                     **/
/**                                                                **/
/** Any access to 0000h through 7FFFh in program memory will clear **/
/** the bank select lines for that operation only. This allows the **/
/** device to jump to interrupt vectors with no intervention, and **/
/** return to the same bank when finished.                         **/
/**                                                                **/
/** PIN DEFINITIONS: These definitions are for 28 pin PLCC and DIP **/

```



```

/** Input pins */
PIN 1  = CLKIN;      Clock input for GAL latches.
PIN 2  = A15;        Microcontroller address lines for address decode.
PIN 3  = A14;
PIN 4  = A13;
PIN 5  = A12;
PIN 6  = A11;
PIN 8  = A10;
PIN 9  = A9;
PIN 10 = A8;
PIN 11 = A7;
PIN 12 = A6;
PIN 13 = A5;
PIN 14 = D3;         Microcontroller data lines to set bank selection.
PIN 15 = D2;
PIN 16 = D1;
PIN 17 = D0;
PIN 28 = WR_STROBE;  Microcontroller write strobe

/** Output pins */
PIN 20 = CLKOUT;      Qualified microcontroller write strobe. It is fed
;                      back into the GAL 26V12 CLK input. This
;                      signal must be assigned to pin 20 or 22 because
;                      it requires 12 product terms.
PIN 18 = A18_LATCH;    Temporary latch of A18 memory signal.
PIN 19 = A17_LATCH;    Temporary latch of A17 memory signal.
PIN 22 = A16_LATCH;    Temporary latch of A16 memory signal.
PIN 23 = A15_LATCH;    Temporary latch of A15 memory signal.

PIN 24 = MEMADR_18;    Output to memory device pin A18.
PIN 25 = MEMADR_17;    Output to memory device pin A17.
PIN 26 = MEMADR_16;    Output to memory device pin A16.
PIN 27 = MEMADR_15;    Output to memory device pin A15.

;PIN 7  = VCC
;PIN 21 = GND

/** Qualify write strobe to detect valid bank latch write. */
BANK_SEL = A15 & A14 & A13 & A12 & A11 & A10 & A9 & A8 & A7 & A6 & A5
CLKOUT = !WR_STROBE & BANK_SEL

/** BANK SELECT 0 GENERATION */
A15_LATCH.D = D0
A15_LATCH.OEMUX = 0 ;Disable external expression of this signal.
MEMADR_15 = A15_LATCH.Q & A15

/** BANK SELECT 1 GENERATION */
A16_LATCH.D = D1
A16_LATCH.OEMUX = 0 ;Disable external expression of this signal.
MEMADR_16 = A16_LATCH.Q & A15

```

```

/** BANK SELECT 2 GENERATION **/
A17_LATCH.D = D2
A17_LATCH.OEMUX = 0 ;Disable external expression of this signal.
MEMADR_17 = A17_LATCH.Q & A15

/** BANK SELECT 3 GENERATION **/
A18_LATCH.D = D3
A18_LATCH.OEMUX = 0 ;Disable external expression of this signal.
MEMADR_18 = A18_LATCH.Q & A15

```

### USING THE ROMSIZE FEATURE

The ROMSIZE feature allows software to dynamically reconfigure program memory size, permitting a portion of program memory to be switched between on- and off-chip. It provides an easy way to increase program memory to 64KB plus on-chip memory. In addition, it simplifies the task of building a boot-loader for external programmable memory, such as FLASH, EEPROM, or Non-Volatile SRAM (NVS RAM).

Using the ROMSIZE feature is very straightforward. Bits RMS2, RMS1, RMS0 (ROMSIZE.2-0) select the maximum amount of on-chip memory. The ROMSIZE select bits are Timed Access protected to ensure maximum software reliability. Any program memory accesses outside of the range defined by the ROMSIZE register will automatically be fetched externally via ports 0 and 2. External code fetches on devices with the ROMSIZE™ feature are performed in the same way as on all members of the High-Speed Microcontroller Family. The designer is reminded that if ports 0 and 2 will be used for external memory access, they should not be used as general purpose I/O ports.

The modification of the ROMSIZE register must be followed by a 2 machine cycle delay, such as executing 2 NOP instructions, before jumping to the new address range. Interrupts must be disabled during this operation, because a jump to the interrupt vector during the changing of the memory map can cause erratic results. The procedure to reconfigure the amount of on-chip memory is as follows:

1. Jump to a location in program memory that will be unaffected by the change,
2. Disable interrupts by clearing the EA bit (IE.7),
3. Write AAh to the Timed Access Register (TA;C7h)
4. Write 55h to the Timed Access Register (TA;C7h)
5. Modify the ROM Size Select bits (RMS2-0),
6. Delay 2 machine cycles (2 NOP instructions),

7. Enable interrupts by setting the EA bit (IE.7).

There are a number of software considerations when using the ROMSIZE feature to switch between on- and off-chip memory. Modification of the ROM Size Select register must be made from a program memory location that will be valid both before and after the on-chip memory configuration. Care must be exercised when assembling or compiling the program so that all the modules are located at the correct starting address, including the interrupt vector table.

If the 0 kbyte on-chip memory option is selected, extra precautions must be taken. It is necessary to duplicate the interrupt vector table in off-chip memory when switching the lower 1KB of program memory from on-chip to off-chip. In general, applications will find it most useful to reduce the on-chip memory no smaller than 1KB. This will maximize the addressable external memory range, while keeping the interrupt vectors on-chip. The 0KB option is most useful when the on-chip memory is only used as a boot loader.

### EXPANDING MEMORY BEYOND 64KB WITH THE ROMSIZE FEATURE

Addressing more than 64KB of external memory in conjunction with the ROMSIZE™ feature is done similar to the ROMless method. The primary difference is that the ROMSIZE™ feature allows the designer to use the on-chip program memory as the "common" block. This simplifies the construction of external hardware, as the common block memory signal (the A15 signal in the examples presented) does not have to be decoded.

The key to designing with the ROMSIZE feature is to incorporate the on-chip memory into the memory map with the most efficient memory utilization and simplest decoding method. There are many approaches to this problem, but only one will be presented here. This example uses 16KB of on-chip memory, plus eight 48KB pages of expanded memory located in a 27C040

512KB EPROM. This provides a total program memory of 400KB. The interrupt vectors and service routines are contained in the on-chip memory for fast access.

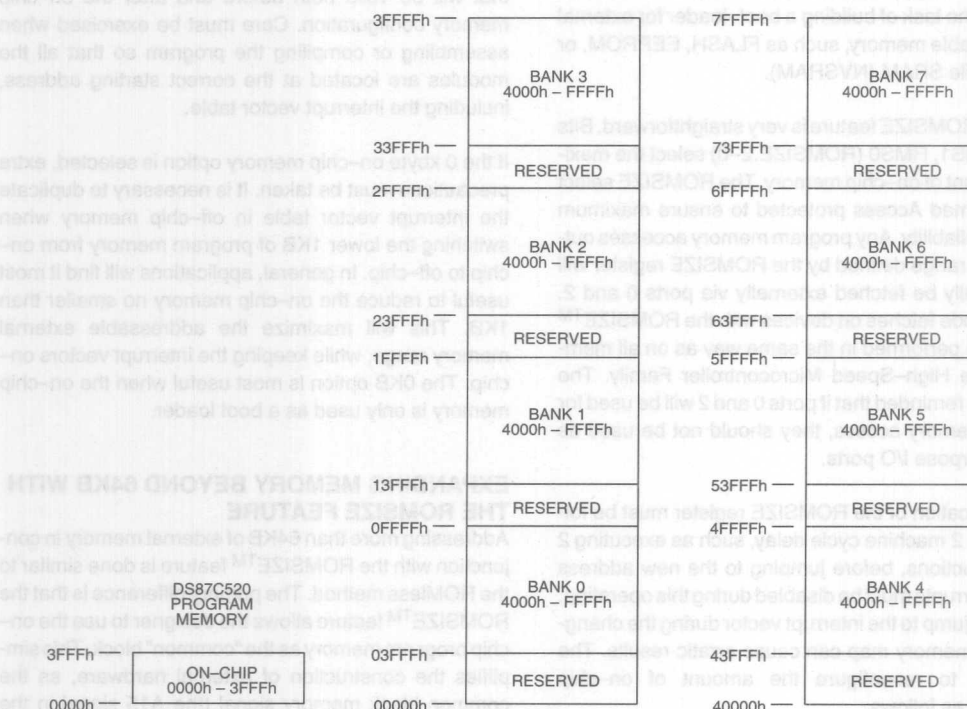
Figure 7 shows one possible memory map for use with the DS87C520 incorporating 16KB of on-chip EPROM. Note that program memory from 0000h to 3FFFh on each external page is not used. This greatly simplifies the design of the memory decode, requiring no external logic and one less I/O line. Figure 8 shows how to use three I/O lines to directly decode the upper address

lines of the device. Software for this configuration is similar to that presented in previous examples.

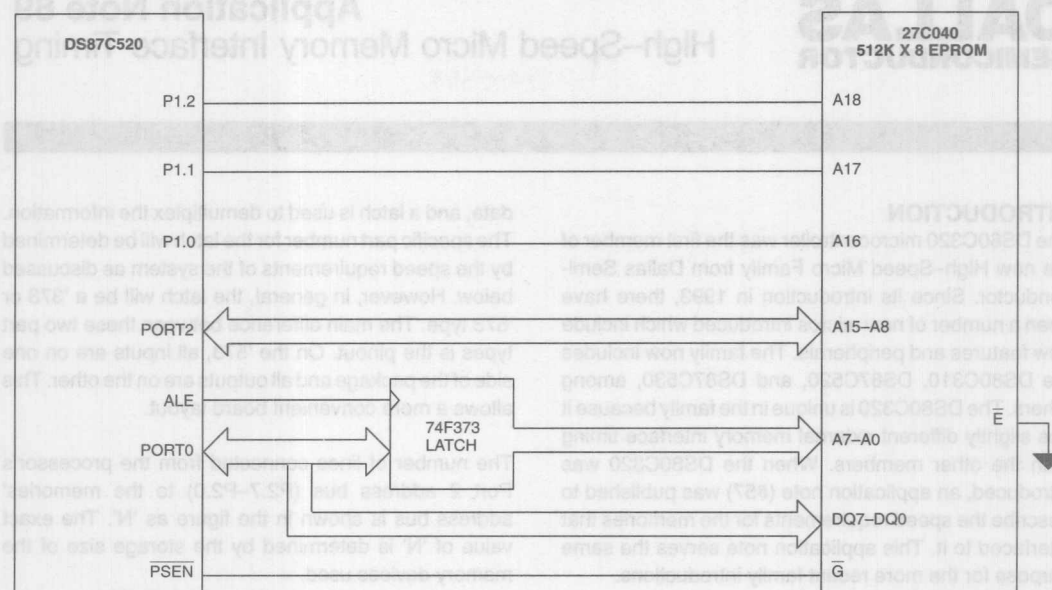
It is also possible to use the ROMSIZE™ feature in conjunction with a latched bank address on the MOVX bus. Similar to the ROMless example, this approach does not require dedicated I/O pins for bank switching. Because the use of on-chip program memory allows for a simpler decode circuit, less expensive PLDs may be used.

## ROMSIZE™ FEATURE COMMON-PAGE EXPANSION MEMORY MAP Figure 7

27C040 EPROM



## ROMSIZE™ FEATURE COMMON-PAGE I/O EXPANSION Figure 8



Expanding the amount of data memory used by the microcontroller is the easiest form of memory expansion. Because there is no possibility of interfering with program execution, timing is not as critical. General purpose I/O lines can be connected directly to the address lines or chip enables of the memory device(s). The

appropriate port pin can be directly modified to access the correct page prior to the memory operation. If the application requires all available I/O lines, then a latched bank address scheme demonstrated in the above examples can be used.

For this reason, the family cannot be used for 33 MHz operation. The other relevant property of the latch is its propagation delay from input to output. Since the latch is in the address path, this parameter has a direct and significant impact on the memory timing requirements. This parameter will be discussed in the following section.

As with all 8051 external memory interfaces, Port 0 lines (P0.7-P0.0) of the processor carry both address and

A common configuration for a High-Speed Micro-based system is shown in Figure 1. In this example, both program (EPROM) and data (SRAM) memory devices are included in the system. Of course with an EPROM-based part such as the DS87C520, it is likely that no other program storage will be required outside the processor. However, for the purpose of this discussion, it will be assumed that external program storage will be used. If the application dictates the use of both on-board and external program memories, some additional decoding logic (not shown) may be required so that the two memory spaces do not overlap.

As with all 8051 external memory interfaces, Port 0 lines (P0.7-P0.0) of the processor carry both address and

**DALLAS**  
**SEMICONDUCTOR**

## Application Note 89

### High-Speed Micro Memory Interface Timing

#### INTRODUCTION

The DS80C320 microcontroller was the first member of the new High-Speed Micro Family from Dallas Semiconductor. Since its introduction in 1993, there have been a number of new micros introduced which include new features and peripherals. The family now includes the DS80C310, DS87C520, and DS87C530, among others. The DS80C320 is unique in the family because it has slightly different external memory interface timing than the other members. When the DS80C320 was introduced, an application note (#57) was published to describe the speed requirements for the memories that interfaced to it. This application note serves the same purpose for the more recent family introductions.

One major difference in the timing of the new members of the family and the DS80C320 is that the original DS80C320 had a maximum clock rate of 25 MHz. All of the more recent 5 volt micros were introduced at a maximum clock rate of 33 MHz. Obviously, this affects the timing of the external memory interfaces significantly. The analysis that follows is based on a "worst case" timing using a 33 MHz clock, but will also identify memory speeds required for other frequencies as well.

A common configuration for a High-Speed Micro based system is shown in Figure 1. In this example, both program (EPROM) and data (SRAM) memory devices are included in the system. Of course with an EPROM based part such as the DS87C520, it is likely that no other program storage will be required outside the processor. However, for the purposes of this discussion, it will be assumed that external program storage will be used. If the application dictates the use of both on-board and external program memories, some additional decoding logic (not shown) may be required so that the two memory spaces do not overlap.

As with all 8051 external memory interfaces, Port 0 lines (P0.7-P0.0) of the processor carry both address and

data, and a latch is used to demultiplex the information. The specific part number for the latch will be determined by the speed requirements of the system as discussed below. However, in general, the latch will be a '373 or '573 type. The main difference between these two part types is the pinout. On the '573, all inputs are on one side of the package and all outputs are on the other. This allows a more convenient board layout.

The number of lines connected from the processor's Port 2 address bus (P2.7-P2.0) to the memories' address bus is shown in the figure as 'N'. The exact value of 'N' is determined by the storage size of the memory devices used.

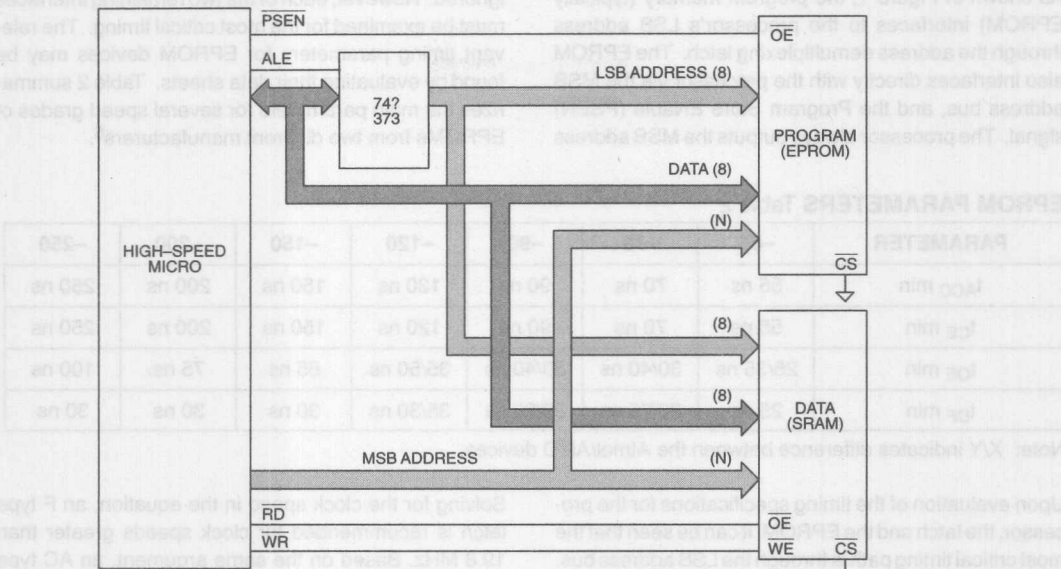
#### LATCH REQUIREMENTS

Due to the high speed of the Port 0 (AD7-AD0) bus, some consideration must be given to the choice of latches used for address demultiplexing. By examining the High-Speed Micro data sheet, it can be seen that some timing constraints are placed on the latch. For instance, the CPU parameter  $t_{AVLL}$  (Port 0 Address Valid to ALE Low) determines the minimum setup time ( $t_{SU}$ ) the latch will actually have. The parameters  $t_{LHL}$  and  $t_{LLAX}$  also affect the timing requirements of the latch. Table 1 shows the CPU parameters for 33 MHz operation, and the requirements placed on various latch families. For the parameters in the table, the CPU parameters must be greater than the latch parameters. It can be seen that the minimum required setup and hold time are violated for the HC latch family (highlighted). For this reason, this family cannot be used for 33 MHz operation.

The other relevant property of the latch is its propagation delay from input to output. Since the latch is in the address path, this parameter has a direct and significant impact on the memory timing requirements. This parameter will be discussed in the following section.



TYPICAL HIGH-SPEED MICRO SYSTEM Figure 1



LATCH PARAMETERS Table 1

CPU PARAMETER	@33 MHz	LATCH PARAMETER	AC FAMILY	F FAMILY	HC FAMILY
t <sub>LLHL</sub> min	40 ns	t <sub>w</sub>	4.5 ns	6.0 ns	20.0 ns
t <sub>AVLL</sub> min	10 ns	t <sub>SU</sub>	6.0 ns	2.0 ns	15.0 ns
t <sub>LLAX</sub> min	10 ns	t <sub>H</sub>	1.0 ns	3.0 ns	13.0 ns
		t <sub>PROP</sub>	11.5 ns	8.0 ns	38.0 ns

## PROGRAM MEMORY

As shown in Figure 1, the program memory (typically EPROM) interfaces to the processor's LSB address through the address demultiplexing latch. The EPROM also interfaces directly with the processor via the MSB address bus, and the Program Store ENable (PSEN) signal. The processor always outputs the MSB address

before the LSB address, so this interface can be ignored. However, each of the two remaining interfaces must be examined for the most critical timing. The relevant timing parameters for EPROM devices may be found by evaluating their data sheets. Table 2 summarizes the main parameters for several speed grades of EPROMs from two different manufacturers<sup>6</sup>.

**EPROM PARAMETERS** Table 2

PARAMETER	-55	-70	-90	-120	-150	-200	-250
$t_{ACC}$ min	55 ns	70 ns	90 ns	120 ns	150 ns	200 ns	250 ns
$t_{CE}$ min	55 ns	70 ns	90 ns	120 ns	150 ns	200 ns	250 ns
$t_{OE}$ min	25/35 ns	30/40 ns	30/40 ns	35/50 ns	65 ns	75 ns	100 ns
$t_{DF}$ min	25 ns	30/25 ns	30/25 ns	35/30 ns	30 ns	30 ns	30 ns

Note: X/Y indicates difference between the Atmel/AMD devices.

Upon evaluation of the timing specifications for the processor, the latch and the EPROM, it can be seen that the most critical timing path is through the LSB address bus. The address must appear on this bus, pass through the latch, address the EPROM, and the EPROM must output valid data in less time than the CPU parameter  $t_{AVIV}$ . Since the latch is in the path, the timing of this bus can be expressed by the following equation:  $t_{PROP} + t_{ACC} < t_{AVIV}$ . The DS87C520 data sheet shows that  $t_{AVIV}$  is a function of clock speed (denoted  $t_{CLCL}$ ), and is given by:  $t_{AVIV} = 3t_{CLCL} - 20$  ns. Solving these equations for 33 MHz operation using an F type latch, it can be seen that an EPROM access time of less than 63 ns is required. Therefore for full speed operation, an EPROM with 55 ns address access time must be used because this is the slowest speed grade that meets the requirement of 63 ns.

The equation above shows that the latch speed directly impacts the required speed of the EPROM. Since a fast latch is less expensive than a fast EPROM, an F type latch is recommended at clock speeds that would necessitate the use of an EPROM of 120 ns or faster.

Solving for the clock speed in the equation, an F type latch is recommended for clock speeds greater than 19.8 MHz. Based on the same argument, an AC type latch is recommended for clock speeds greater than 16.8 MHz.

Table 3 shows the EPROM speeds and latch types recommended for various CPU clock speeds. The suggested speed grade is based on the above equation and the EPROM and latch timing parameters. Further evaluation shows that the EPROM parameter  $t_{DF}$  may also be a critical parameter at some high CPU clock speeds. This parameter must always be less than the CPU parameter  $t_{PXIZ}$ . As indicated in Table 2,  $t_{DF}$  varies for the same speed grade device from different manufacturers. Therefore in Table 3, AMD is the recommended manufacturer where a CPU frequency of 29.4912 MHz is used. If the Atmel device were used, the  $t_{PXIZ}$  parameter would be violated. This exception applies to any clock frequency between 30.61 MHz where the switch of a 70 ns EPROM is made and 28.47 MHz where processor requirement for  $t_{PXIZ}$  becomes 30 ns.

6. EPROM devices from AMD and Atmel were considered.

**RECOMMENDED EPROM SPEEDS** Table 3

CLOCK FREQUENCY (MHz)	SPEED WITH 'F373 LATCH	SPEED WITH 'AC373 LATCH	SPEED WITH 'HC373 LATCH
33.0	55	55	N/A
29.4912	70*	70	N/A
25.0	90	70	55
22.1184	90	90	70
20.0	120	90	90
19.8	120	120	90
18.432	120	120	90
16.8	150	120	120
16.0	150	150	120
14.746	150	150	120
14.318	150	150	150
12	200	200	150
11.059	200	200	200
7.37	250	250	250
1.8432 and below	250	250	250

\* : Should be AMD 70 ns device because of  $t_{DF}$ .

### DATA MEMORY

There are a number of factors that make interfacing data memories (SRAMs) to the High-Speed Microcontroller family extremely easy. First, SRAM devices are generally faster, and more readily available in higher speed grades. In fact, it is sometimes difficult to find a slow SRAM. A more significant factor is that all High-Speed Micro Family members have the ability to insert stretch cycles into the MOVX instructions. This provides a convenient means of supporting both high and low speed devices on the same data bus without requiring external support hardware. All High-Speed Micro Family members default to the use of one stretch cycle for MOVX instructions. To obtain maximum throughput, application software can write to certain Special Function Register (SFR) bits and cause the MOVX instructions to operate with zero stretch cycles. This default condition is a convenience to existing designs that may not have fast RAM in place. Even in high speed systems, it may not be necessary or desirable to perform data accesses at full speed. Additionally, there are a variety of memory

mapped peripherals such as LCD displays or UARTs that are not fast enough to keep up with the full speed high-speed micro. This flexibility allows the user to trade some performance for slower data RAMs if so desired. For the discussion that follows, a worst case timing scenario of zero stretch cycles will be assumed.

For maximum performance, i.e., with a zero stretch cycle data memory access programmed into the processor, a MOVX instruction requires two machine cycles. The fetch of the instruction takes one machine cycle leaving one machine cycle for the memory read or write. In the analysis of the data memory's timing requirements that follows, it will be assumed that the recommendations of Table 3 have been followed. Specifically, this means that an "F" family latch is used for CPU clock frequencies greater than 19.8 MHz, an "AC" family latch is used for frequencies greater than 16.8 MHz, and an "HC" family latch is used for lower frequencies.

Through analysis and a survey of memory products<sup>7</sup>, it can be determined that four SRAM timing parameters are necessary and sufficient to meet the processor's timing requirements for most situations. These parameters and their values for various speed grades are shown in Table 4. During a data read operation, the processor expects the time from an address change until valid data is available to be  $71 \text{ ns}$  ( $t_{\text{AVDV1}} = 3t_{\text{CLCL}} - 20$ ) or less. If the propagation delay from D to Q of an F373 latch (8 ns) is subtracted from this parameter, you obtain a memory address access ( $t_{\text{AA}}$ ) requirement of 63 ns. Also for a data read operation, the time from the  $\overline{\text{RD}}$  signal going low until valid data is received from the memory device must be 41 ns ( $t_{\text{RLDV}} = 2t_{\text{CLCL}} - 20$ ) or less. Since the processor's  $\overline{\text{RD}}$  signal is tied to the memory's  $\overline{\text{OE}}$  pin, the memory must have an output enable access time ( $t_{\text{OE}}$ ) of less than 41 ns. After the processor has read the data, the SRAM must relinquish the bus within 25 ns ( $t_{\text{RHDZ}} = t_{\text{CLCL}} - 5$ ). This dictates that the SRAM parameter  $t_{\text{OHZ}}$  be less than 25 ns. For a write, the processor will provide a minimum write pulse of 56 ns ( $t_{\text{WLWH}} = 2 t_{\text{CLCL}} - 5$ ), which is equal to the minimum required write pulse width ( $t_{\text{WP}}$ ) of the SRAM. On the basis of these four calculated parameters and assumed SRAM speeds shown in

Table 4, the appropriate speed device may be determined for a number of different clock frequencies. A summary of the recommended RAM speeds is given in Table 5. It should be noted that the critical timing parameter is not always the access time. Because of the high speed of the processor and variations in memory parameter relationships, all four parameters must be checked for any specific clock speed.

**SRAM PARAMETERS** Table 4

$t_{\text{AA}}$ (ns)	$t_{\text{OE}}$ (ns)	$t_{\text{OHZ}}$ (ns)	$t_{\text{WP}}$ (ns)
35	20	15	25
55	30	25	35
70	35	30	45
80	35	30	60
100	50	35	60
120	60	45	70
150	55	40	90
170	80	35	120
200	100	35	150

**RECOMMENDED RAM SPEEDS** Table 5

CLOCK (MHz)	LATCH	MEMORY SPEED (zero stretch)	MEMORY SPEED (one stretch)
33.0	F373	55 ns	150 ns
29.4912	F373	55 ns	170 ns
25.0	F373	80 ns	170 ns
22.1184	F373	100 ns	200 ns
20.0	F373	120 ns	200 ns
19.8	AC373	120 ns	200 ns
18.432	AC373	120 ns	200 ns
16.8	HC373	120 ns	200 ns
16.0	HC373	120 ns	200 ns
14.746	HC373	120 ns	200 ns
14.318	HC373	150 ns	200 ns
12	HC373	170 ns	200 ns
11.059	HC373	200 ns	200 ns
7.37	HC373	200 ns	200 ns
1.8432 and below	HC373	200 ns	200 ns

7. Memory data books from Dallas Semiconductor 1992-93, Fujitsu 1990, Hitachi #M18, Micron 1992, Mosel 1991-92, NEC 1989, Sony 1991 were surveyed for suitable SRAM products.

## ADDITIONAL CONSIDERATIONS

All of the timing calculations used in this application note are based on equations found in the DS87C520 data sheet. These specifications assume an approximately equal capacitive load on the signals specified. If the configuration of Figure 1 is used, this is achieved. If, however, any signal is connected to additional loads, then the capacitive loading including the additional devices should be evaluated. If there is a significant difference, additional margins should be used in the critical path analysis, and appropriate memory speeds selected.

## EPROM Equations

### PSEN Access

$$\begin{aligned} t_{OE} &= t_{PLIV} \\ &= 2t_{CLCL} - 20 \end{aligned}$$

### Address Access

$$\begin{aligned} t_{ACC} &= t_{AVIV} - \text{latch delay} \\ &= 3t_{CLCL} - 20 - \text{latch delay} \\ &\quad (\text{same for RAM}) \end{aligned}$$

### Bus Release

$$\begin{aligned} t_{DF} &= t_{PXIZ} \\ &= t_{CLCL} - 5 \end{aligned}$$

## RAM Equations

### Read Access

$$t_{OE} = t_{RLDV}$$

$$\begin{aligned} &(\text{zero stretch}) \\ &= 2t_{CLCL} - 20 \end{aligned}$$

$$\begin{aligned} &(\text{one stretch}) \\ &= 4t_{CLCL} - 20 \end{aligned}$$

### Write Pulse

$$t_{WP} = t_{WLWH}$$

$$\begin{aligned} &(\text{zero stretch}) \\ &= 2t_{CLCL} - 5 \end{aligned}$$

$$\begin{aligned} &(\text{one stretch}) \\ &= 4t_{CLCL} - 10 \end{aligned}$$

### Bus Release

$$t_{OHZ} = t_{RHDZ}$$

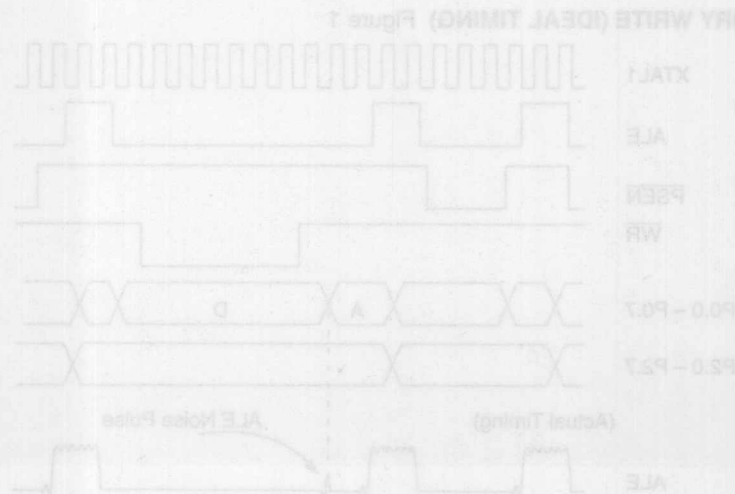
$$\begin{aligned} &(\text{zero stretch}) \\ &= t_{CLCL} - 5 \end{aligned}$$

$$\begin{aligned} &(\text{one stretch}) \\ &= 2t_{CLCL} - 5 \end{aligned}$$

For older or otherwise unconventional SRAM devices, it may be wise to confirm other important timing parameters (such as data setup before write active). However, on the devices surveyed, meeting the four parameters discussed above will qualify the device for use.

## EQUATION SUMMARY

For the user who wishes to calculate the memory speed requirements using a crystal frequency not shown in the preceding tables, the following equations provide a concise summary of the information needed.





# DALLAS SEMICONDUCTOR

## Application Note 91 Microcontroller Design Guidelines for Reducing ALE Signal Noise

### OVERVIEW

The 8051 architecture allows for external program and data access through the use of Port 0 and Port 2 as an external memory interface. The 8051 architecture multiplexes the data and LSB of address on Port 0, requiring a 74373 latch for demultiplexing. This means that Port 0 will be directly connected to at least two devices. More devices may be placed on the bus if an external data SRAM or memory-mapped peripherals are used.

Because Port 0 must switch quickly between address and data, it requires strong current drive characteristics. The need to quickly switch many loads requires strong drive characteristics on Port 0. Unfortunately, the high instantaneous current requirements of quickly switching all the pins of Port 0 can induce noise on the ALE signal. In some instances, this noise can interfere with program and data accesses by causing the external hardware to latch an incorrect address. This is a relatively rare occurrence, and most designers will not encounter it. The magnitude of this problem is directly related to several issues associated with both the system and software. Devices which do not access external memory via Port 0 and Port 2 will not experience this problem.

This application note will discuss ways the system designer can reduce the effects of Port 0 switching on device operation. It is applicable to any ROMless 8051 microcontroller which accesses external memory via Port 0 and Port 2, including the DS80C310 and DS80C320. It is also applicable to any microcontroller with internal program memory that accesses external memory.

### ALE NOISE GENERATION

Under certain system conditions, noise induced on ALE can cause an incorrect LSB address to be latched when using the multiplexed address/data bus. The noise, as seen in Figure 1, is generated by the high speed switching of Port 0 when the processor stops driving a memory address and begins driving data during a MOVX write. The noise pulse can, under the right conditions, rise above the  $V_{IH}$  input threshold of TTL, LS, FS and HCT logic. In this case, the 74373 latch may be falsely triggered, latching an incorrect address and disturb the LSB address of the MOVX write.

**DATA MEMORY WRITE (IDEAL TIMING) Figure 1**

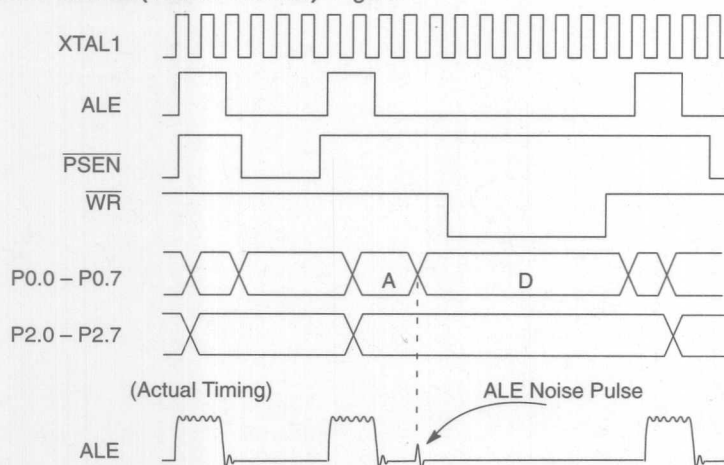
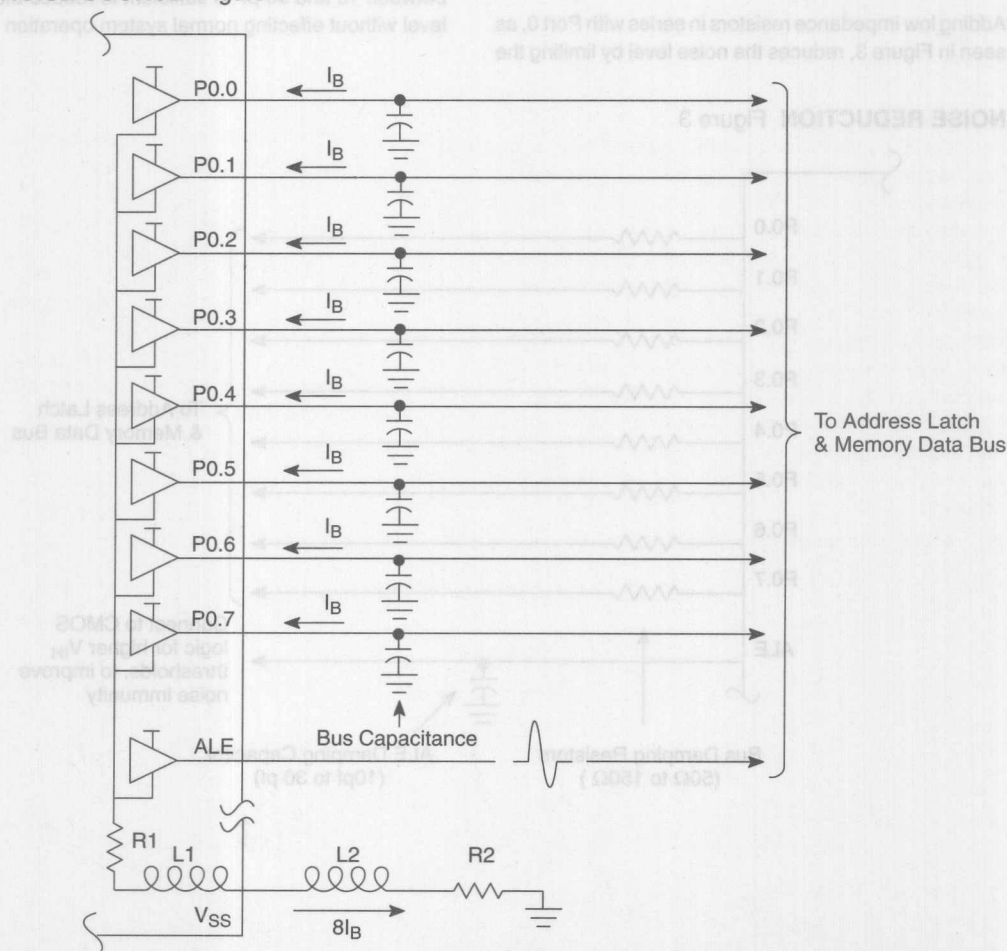


Figure 2 shows a system diagram of how the noise pulse is generated. The noise pulse is produced when the processor drives a Port 0 pin with a high address (see "A" in Figure 1) followed by a low for data (see "D" in Figure 1). The device must sink a relatively large amount of current on each pin ( $I_B$ ) to take the line from a high to a low state. It is obvious that the more pins which change from a high to a low, the larger the noise. The worst case will be during a MOVX write instruction with an LSB address of FF (Hex) and a data byte of 00 (Hex). Because all eight port pins are switching simultaneously, the maximum amount of current will be drawn into the microcontroller. The combined inductance and

resistance both inside the processor and in the system result in the processor internal ground rising above the system ground. This in turn induces the noise seen on ALE. The case of a MOVX read does not involve the sinking of current by the processor and should not induce significant noise on the ALE signal. System elements which have a direct relationship to the magnitude of the noise are:

1. Port 0 bus capacitance
2. System ground inductance ( $L_2$ ) and resistance ( $R_2$ )
3. System supply voltage ( $V_{CC}$ )

**ALE NOISE SOURCE** Figure 2



mize the effect of Port 0 switching on ALE noise. Reducing bus capacitance reduces the energy required to be discharged which results in lower peak currents and reduced peak voltages in the noise pulse. Reducing the external ground resistance and inductance also reduces the noise level, by reducing the resistive and inductive voltage drop.

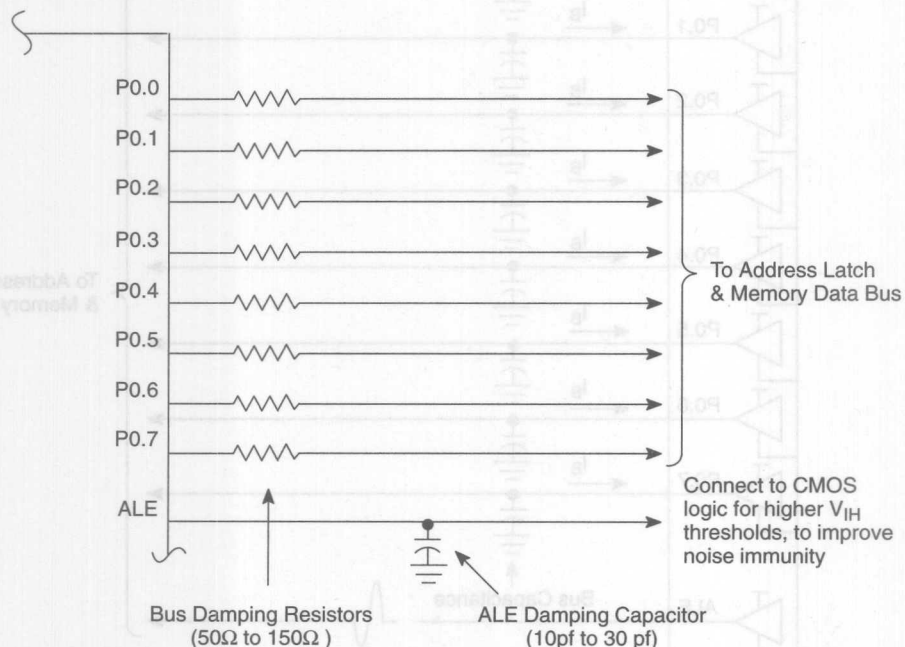
The supply voltage is also directly proportional to the voltage level of the noise pulse. Maintaining  $V_{CC}$  within recommended specifications will limit the noise voltage level.

Adding low impedance resistors in series with Port 0, as seen in Figure 3, reduces the noise level by limiting the

effect the slew rate or final input voltage level to the memory as the processor writes to external memory. Values in the range of  $50\Omega$  to  $150\Omega$  can generally be used without disturbing write cycle times. Actual values for the series resistance should be verified in the end system.

Use of a capacitor on the ALE signal line will also significantly reduce the noise pulse. Again, values must be verified in the system, with care used in not reducing the slew rate of the ALE signal to a point that memory access is no longer valid. Generally a capacitance of between 10 and 30 pF is sufficient to reduce the noise level without effecting normal system operation

**NOISE REDUCTION** Figure 3



## INPUT THRESHOLDS

The simplest and most reliable method of eliminating the address latch related noise is to select a logic family with a high input threshold. Standard TTL, LS, FS, and HCT logic parts have a  $V_{IH}$  threshold of approximately 2.0 volts. HC (High-Speed CMOS) or AC (Advanced CMOS) logic, on the other hand, has a  $V_{IH}$  of approximately 3.5 volts at a supply voltage of 5 volts. The higher threshold level of the HC or AC CMOS logic increases the noise immunity by approximately 1.5 volts. This is generally all that is needed to prevent the undesired latching by ALE.

One disadvantage of using CMOS logic is that it is slower than other logic families. Propagation delays through CMOS logic are generally in the range of 18 ns for HC and 10 ns for AC, compared with 2 to 4 ns for FS logic when using a supply of 5 volts. For slower microcontrollers such as the DS5000, DS5001, and DS5002, the propagation delay is usually not an issue because of the slow clock rate. Faster microcontrollers such as the High-Speed Microcontrollers should carefully consider the timing effects of slower logic. In any event, testing should be done in the final application to verify the effects using slower CMOS logic.

One disadvantage of using CMOS logic is that it is slower than other logic families. Propagation delays through CMOS logic are generally in the range of 15 ns for HC and 10 ns for AC, compared with 5 to 4 ns for PS logic when using a supply of 5 volts. For slower microcontrollers such as the DS5000, DS6001, and DS6002, the propagation delay is usually not an issue because of the slow clock rate. Faster microcontrollers such as the High-Speed Microcontrollers should carefully consider the timing effects of slower logic. In any event, testing should be done in the final application to verify the effects using slower CMOS logic.

## INPUT THRESHOLDS

The simplest and most reliable method of eliminating the address latch related noise is to select a logic family with a high input threshold. Standard TTL LS, FS, and HCT logic parts have a  $V_{IH}$  threshold of approximately 2.0 volts. HC (High-Speed CMOS) or AC (Advanced CMOS) logic, on the other hand, has a  $V_{IH}$  of approximately 3.5 volts at a supply voltage of 5 volts. The higher threshold level of the HC or AC CMOS logic increases the noise immunity by approximately 1.5 volts. This is generally all that is needed to prevent the undesired latching by ALE.



## DEVELOPMENT SUPPORT

Dallas Semiconductor has a wide range of services designed to support its customers. Microcontroller applications engineers are available Monday through Friday to provide technical support from 8 am to 5 pm Central Standard Time. They can be reached by telephone and electronic mail. In addition, technical data sheets and application notes are available from the Dallas Semiconductor Fax-On-Demand service 24 hours per day. The service is operated by entering responses on a Touch Tone phone, and users should first obtain a copy of the index before ordering specific documents.

Dallas Semiconductor operates a bulletin board system which contains software examples for many of its products. The modem operates at standard baud rates up to a maximum rate of 14,400, with the setting of no parity. 8 data bits. It can be accessed using standard communication software.

Application Support: (214) 450-5769 (voice)  
(214) 450-5715 (fax)  
Fax-On-Demand: (214) 450-0441  
Bulletin Board: (214) 450-5769  
E-mail: micro.support@dallasmi.com

Dallas Semiconductor also maintains a presence on the Internet, with both a World Wide Web home page and anonymous ftp site. The home page has access to company information, data sheets, application notes, and product information. The ftp server contains data sheets and application notes.

World Wide Web home page: <http://www.dallasmi.com>  
Anonymous FTP: <ftp.dallasmi.com>

### DEVELOPMENT TOOLS

The following incomplete list of High-Speed Microcontroller development tool vendors is provided as a starting point.

view to our customers to assist them in locating sites for use with Dallas Semiconductor microcontroller products. Dallas Semiconductor neither recommends, warrants, nor provides technical support for any product not manufactured by Dallas Semiconductor. The inclusion or exclusion of any vendor from this list is in no way a reflection of the vendor or the product.

### SOFTWARE COMPATIBILITY

Dallas Semiconductor microcontrollers execute the 8051 instruction set and are opcode compatible with other 8051-based products. The special features of Dallas Semiconductor microcontrollers are accessed via Special Function Registers unique to our products, but the devices do not use any new instructions. The new Special Function Registers can be easily defined in the user's software with EQUATE statements or setup like C code defined, these new Special Function Registers receive the same treatment as any of the original 8051 registers. This means that Dallas Semiconductor microcontrollers are compatible with almost every 8051-based software tool available.

### HIGH-LEVEL LANGUAGE COMPILERS

Like assembly, compilers must be informed of the existence and location of the Special Function Registers unique to Dallas Semiconductor microcontrollers. When using C, it is commonly necessary to identify the starting address for various read/write segments such as XDATA and STACK.

In addition, it is recommended that the large memory model be used in conjunction with C compilers. This places the stack in off-chip SRAM. Microcontroller systems usually have an abundance of such SRAM compared to PC-based systems. While off-chip stack results in slower execution time, the stack size becomes virtually unlimited.

## DEVELOPMENT SUPPORT

### TECHNICAL SUPPORT

Dallas Semiconductor has a wide range of services designed to support its customers. Microcontroller applications engineers are available Monday through Friday to provide technical support from 8 am to 5 pm Central Standard Time. They can be reached by telephone and electronic mail. In addition, technical data sheets and application notes are available from the Dallas Semiconductor Fax-On-Demand service 24 hours per day. The service is operated by entering responses on a Touch Tone phone, and users should first obtain a copy of the index before ordering specific documents.

Dallas Semiconductor operates a bulletin board system which contains software examples for many of its products. The modem operates at standard baud rates up to a maximum rate of 14.4 kb/s, with the settings of no parity, 8 data bits, one stop bit. It can be accessed using standard communication software.

Application Support:	(214) 450-8169 (voice)
	(214) 450-3715 (fax)
Fax-On-Demand:	(214) 450-0441
Bulletin Board:	(214) 450-8169
E-mail:	micro.support@dalsemi.com

Dallas Semiconductor also maintains a presence on the Internet, with both a World Wide Web home page and anonymous ftp site. The home page has access to company information, data sheets, application notes, and product information. The ftp server contains data sheets and application notes.

World Wide Web home page:	<a href="http://www.dalsemi.com">http://www.dalsemi.com</a>
Anonymous FTP:	<a href="ftp.dalsemi.com">ftp.dalsemi.com</a>

### DEVELOPMENT TOOLS

The following incomplete list of High-Speed Microcontroller development tool vendors is provided as a ser-

vice to our customers to assist them in locating aids for use with Dallas Semiconductor microcontroller products. Dallas Semiconductor neither recommends, warrants for suitable use, nor provides technical support for any product not manufactured by Dallas Semiconductor. The inclusion or exclusion of any vendor from this list is in no way a reflection of the vendor or the product.

### SOFTWARE COMPATIBILITY

Dallas Semiconductor microcontrollers execute the 8051 instruction set and are object code compatible with other 8051-based products. The special features of Dallas Semiconductor microcontrollers are accessed via Special Function Registers unique to our products, but the devices do not use any new instructions. The new Special Function Registers can be easily defined in the user's software with EQUATE statements or setup file. Once defined, these new Special Function Registers receive the same treatment as any of the original 8051 registers. This means that Dallas Semiconductor microcontrollers are compatible with almost every 8051-based software tool available.

### HIGH-LEVEL LANGUAGE COMPILERS

Like assemblers, compilers must be informed of the existence and location of the Special Function Registers unique to Dallas Semiconductor microcontrollers. When using C, it is commonly necessary to identify the starting address for various read/write segments such as XDATA and STACK.

In addition, it is recommended that the large memory model be used in conjunction with C Compilers. This places the stack in off-chip SRAM. Microcontroller systems usually have an abundance of such SRAM compared to ROM based systems. While off-chip stack results in slower execution time, the stack size becomes virtually unlimited.

## Integrated Debugging System/LC (IDS/LC)

- ☐ Combines the functions of an In-Circuit Emulator, 128K PROM emulator, editor, and C source-level symbolic and object code debugger in a Borland Turbo C-like environment.
- ☐ Works with 8051 C compilers from Franklin/ Keil, Archimedes, IAR, Avocet, Intermetrics/ Cosmic/ Whitesmith, and 2500AD.
- ☐ Provides real-time operation with no additional wait states.
- ☐ Single Target Interface Unit supports 8051 family variants up to 20 MHz without adapters or pods. Our high speed option provides support of 33 MHz processors.
- ☐ Offers 1,000 source or assembly level breakpoints.
- ☐ Provides complete control of program execution with Run, Reset, Run to Cursor, Step and Stop commands.
- ☐ Has 30 day money back guarantee and one year warranty.
- ☐ The price for a complete 8051 system is just \$995 and is available *now*.

### Contact:

Cactus Logic  
 14120 East Live Oak Avenue, Suite A2  
 Baldwin Park, CA 91706-1345 USA  
 Tel: (800) 847-1998 (818) 337-4547  
 Fax: (818) 337-0689



### Product Information

Cactus Logic's low cost, real-time emulator, the Integrated Debugging System/LC (IDS/LC) provides a PC-based environment for 8051 firmware debugging and testing through your target system's PROM socket. Direct support for Dallas's 80C320 processor. View and modify new special function registers added to the 80C320.

You control the debugging, editing, compiling, and object code downloading functions using simple keystrokes or mouse clicks along with pull-down menu commands and dialog boxes.

The command menus control the operation of the IDS and can be activated with either the keyboard or the mouse. Hot keys are provided for most commands.

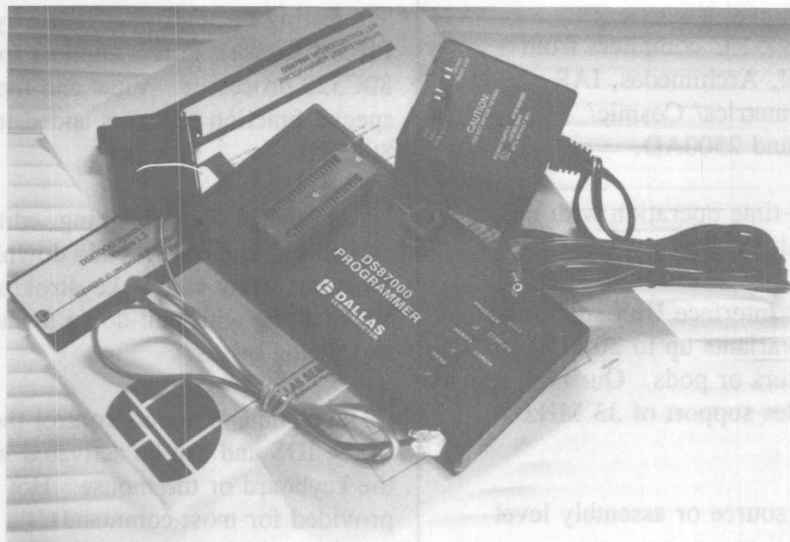
Up to 1,000 powerful source and assembly level breakpoints are simple to set. They can be assigned to groups that can be enabled or disabled by other breakpoints to provide complex triggering. Breakpoints can also provide simulated input or output and logging of registers and variables.

Many other variants of the 8051 are also supported, including those manufactured by Intel, Philips, and Siemens.

The IDS/LC also supports the Motorola 68HC11, Zilog Z80/Z180, and Rockwell/WDC 6502 microprocessor families.

# DALLAS SEMICONDUCTOR

## DS87000 Microcontroller Programmer



### FEATURES

- Operates in stand alone mode or with a PC host
- Easy to use pull down menus in PC hosted mode
- Operation from front panel in stand alone mode
- Used efficient "intelligent" programming techniques
- Non-volatile data and operating parameter storage
- Optional automatic device serialization (Unique identifier written to every device programmed)
- Supports 40-pin DIP packages; plug in adapters available for other package types
- Includes 110 Volt (DS87000-000) or 220 Volt (DS87000-220) power supply

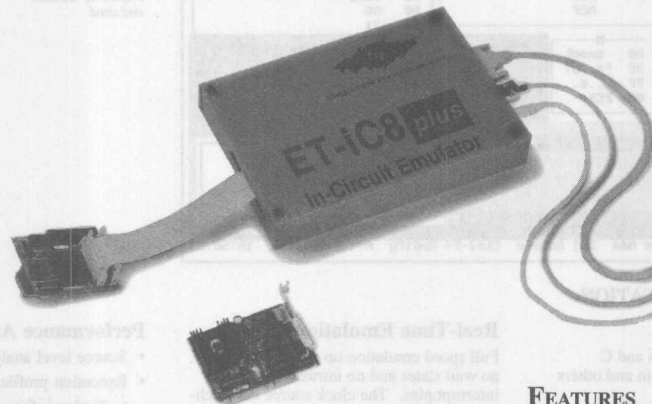
The DS87000 is a convenient, inexpensive method of programming Dallas Semiconductor's EPROM-based microcontrollers. The system comes complete with programmer hardware, menu-based PC software, power supply, and RS-232 cable to connect the unit to a PC serial port. The programmer may be operated with a PC or in stand alone mode. Initially, a PC is used to load programming information into the DS87000. Once loaded, there is no further requirement for the PC or other equipment, allowing it to be easily moved to where it is needed. The DS87000 has the ability to write a unique identification number (serial number) to every device programmed. This feature is found only in more expensive programming equipment.

When operated from the stand alone mode, the DS87000 operates completely from its front panel. Power may be removed from the programmer while it is transported without losing the program buffer. The convenient plug-in transformer provided with the programmer supplies all necessary power.

When operated in PC mode, the DS87000 is operated from the menu-based software provided. The full flexibility of the programmer is available through a system of pull-down menus presented on the PC's screen. The menus are logically arranged to provide convenient access to each command.

# ET-iC8plus Real-Time In-Circuit Emulator

## Supports Dallas 80C320



### DESCRIPTION

The ET-iC8plus is a high performance real-time in-circuit emulator for the Dallas 80C320 microprocessor. Its optimized, integrated environment provides for more efficient development of embedded systems.

### Development Environment

The development environment includes a SAA standard user interface, a debugger, a multi-file editor, and a powerful, integrated project management tool. A flexible interface complements third party C compilers, assemblers, and linkers. The environment can be configured with 1MB of overlay RAM and 1M breakpoints, a trace option, and a high-speed PC link.

### User Interface

Whether you are an experienced or infrequent user, you will find ET-iC8plus' SAA standard user interface friendly and easy to use. It offers pull-down menus, mouse support and context-sensitive on-line help. You can create powerful macros using the complete command set. Hardware jumpers and switches have been eliminated. The selection of clock rate, clock source, VCC target, and reset from target are all controlled via the user interface.

The source level debugger screen window can display either C source code, the corresponding assembly code or both. In the watch window, watch points are used to monitor critical variables. Local or global variables can be displayed or altered in the format declared.

### Breakpoint Flexibility

The intelligent, real-time breakpoints can be set to stop the real-time execution of the program on fetch, read or write, at a single address or within a large range.

### Trace Option

A powerful trace option is available for the ET-iC8plus. It includes high-level support and a performance analyzer. The trace buffer is 32K x 64-bit.

The trace allows sampling of the following signals within the first 256KB:

- 20 address signals
- 8 data signals
- 8 control signals
- 12 auxiliary signals
- a 16-bit absolute or relative time stamp

Together with the 8-bit prescaler, the trace can measure events from nanoseconds to 10 seconds. High-level trace and program performance analysis are clearly presented. A trace range extension to 1MB is optional.

### Maximum Flexibility

The ET-iC8plus base unit will also support a variety of Zilog, Intel, Hitachi, and Toshiba Z80<sup>®</sup> CPU/Z180<sup>™</sup> MPU based microcontrollers. All that is required is a pod change and software.

### FEATURES

#### Emulator

- Real-time emulation up to 20MHz
- 256KB/1MB overlay RAM
- 256K/1M breakpoint RAM
- Intelligent breakpoints
- Background interrupt mode
- Trace option 32K x 64-bit

#### PC Link

- COM x: up to 57.6K baud
- LPT x: up to 156KB/sec
- Optional high-speed, 8-bit, bidirectional, parallel interface

#### SAA Standard User Interface

- On-screen editing
- Pull-down menus
- On-line help
- Programmable function keys
- Mouse and keyboard support
- Powerful macros
- Software switching eliminates hardware jumpers and switches

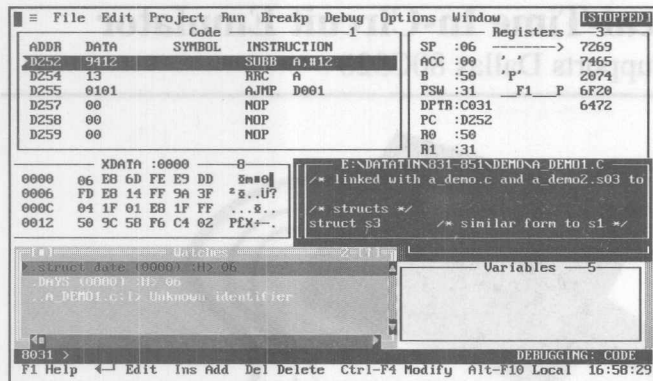
#### Integrated Environment

- Project management
- Multiple file editor
- High-level debugging for C code with watches
- Interfaces with third party tools



EMULATION TECHNOLOGY, INC.  
2344 WALSH AVENUE • BUILDING F  
SANTA CLARA, CA 95051-1301  
TEL 408 / 982.0660 • FAX 408 / 982.0664





### ET-IC8plus User Interface

Multiple windows  
can be selectively  
displayed, located,  
and sized.

## SYSTEM SPECIFICATION

### Language Support

- Third party assemblers and C compilers from Franklin and others

### Source Level Debugging

- Window for source level debugging
- Single step or line step with breakpoints marked directly in the code
- Full support for local and global variables

### Symbolic Support

- Full symbolic debugging and type checking
- Same symbols can be used in different modules

### File Formats

- Archimedes/IAR (UBROF), Tektronix, Motorola S, Intel Hex, Keil/Franklin and PL/M-51

### Fully Integrated Environment

The powerful integrated environment offers a project manager, multiple file editor, high-level debugger easily interfaces with third party C compilers, assemblers, and linkers. The automated test session utilizes powerful macro commands.

### Emulation Memory

256KB overlay RAM can be divided into four banks of 64KB, located anywhere within the 1MB address range of the CPU, mappable in 4KB steps either to the target or the emulator. A 1MB overlay RAM option is available. Memory areas can be write protected.

### Real-Time Emulation

Full speed emulation up to 20MHz means no wait states and no intrusion on I/O or interrupt pins. The clock source is switchable between the target system and the on-board programmable oscillator. The clock rate of the on-board oscillator is software selectable between 1MHz and 16MHz in steps of 1KHz.

### Breakpoints

256K breakpoints (1M optional), with break before executing the instruction (data read and write), combined with intelligent break conditions on single addresses, ranges or high-level language statements provide flexibility.

### Background Interrupt Mode

In the background interrupt mode, non-maskable interrupts (NMI) and all other interrupts are handled even if the real-time emulation ceases. This mode can be enabled or disabled separately for NMI and all other interrupts.

### Trace Option

The trace offers a 32K x 64-bit selective trace buffer, 20-bit address, 8-bit data, 8-bit control, 12 auxiliary signals, 16-bit timer with 8-bit prescaler, complex trigger condition A, B, C with AND, OR, THEN relationship, and 12-bit loop counter. The trace range is 256KB (1MB optional). The trace trigger can be used to generate complex breakpoints.

Trace data are displayed in disassembled format, disassembled with bus state, or in high-level statements.

### Performance Analyzer

- Source level analysis
- Execution profile:
  - Timing information for statements
  - Timing information for functions
  - Callers
  - Average time spent in a program area

### Operating System

#### Personal Computer Support

Minimum: 286 PC, 2MB RAM, 3MB disk space, MS-DOS® 3.1.  
Recommended: 386 PC/33MHz, 4MB RAM, Monochrome, CGA, EGA or VGA.

### Complete Package

The ET-IC8plus emulator comes complete with base unit, external power supply, and PC interface cable, ready to connect to your personal computer. Software with source level debugger and manual are included.

80C320		
Base Unit	Emulator Speed	Max Crystal Speed
ET-IC81001-2	20MHz	20MHz
Item Number	Description	
ET-IC81025	Pod for 80C320	
ET-IC81004-2	Trace (1MB, 20MHz)	
Adapters are available for all pod to target board interconnections.		

© 1994 Emulation Technology, Inc. The Emulation Technology logo is a trademark of Emulation Technology, Inc. Z80 is a registered trademark of Zilog, Inc. Z180 is a trademark of Zilog, Inc. MS-DOS is a registered trademark of Microsoft Corp.



EMULATION TECHNOLOGY, INC.  
2344 WALSH AVENUE • BUILDING F  
SANTA CLARA, CA 95051-1301  
TEL 408 / 982.0660 • FAX 408 / 982.0664

For a FREE Demo Disk  
Call 408-982-0660 Ext.118



## PRODUCT DESCRIPTION

The **iceMASTER-RA** in-circuit emulator uses MetaLink's patented AET technology to provide high performance emulation of Dallas Semiconductor's High-Speed Microcontroller family. Designed for demanding projects, the **iceMASTER-RA** supports frequencies up to 33 MHz with 64K of emulation memory, external data memory and a transparent trace buffer 16K x 32 bit frames deep with advanced searching capabilities. The emulator can operate in a stand-alone mode as well as plugged directly into the target application.

The **iceMASTER** windowed user interface delivers the highest development productivity. Its context sensitive hypertext and hyperlinked HELP system makes this interface easy to learn and easy to use. This powerful, productive interface gives you total control and flexibility in the configuration of the size, position, content and color of each window.

The **iceMASTER** includes a full symbolic and source-level debugger for Assemblers and Compilers. The emulator supports the most popular 8051 Assemblers and Compilers.

## KEY CHARACTERISTICS

- ☐ Supports the **DALLAS High-Speed Microcontroller family** of devices up to 33MHz.
- ☐ Supports **DS80C320, DS87C520 and DS87C530**
- ☐ Full-Featured, Real-time & Transparent Emulator
- ☐ Consists of emulator base with interchangeable probe cards
- ☐ Plugs directly into target applications or operates in a stand-alone mode.
- ☐ Based on patented AET design architecture

## HARDWARE CAPABILITIES

- ☐ Up to 64K Program & 64K External Data Memory
- ☐ 16K x 32-bit frame trace buffer
- ☐ View trace while executing
- ☐ Up to 128K hardware breakpoints
- ☐ Up to 64K Trace ON & 64K Trace OFF triggers
- ☐ Integrated diagnostic self-test capability
- ☐ Interchangeable Probe cards

## SYSTEM FEATURES

- ☐ PC-hosted via RS-232 serial link
- ☐ Efficient, powerful, easy to learn
- ☐ User control of window size, content & color
- ☐ Supports third party Assemblers & Compilers
- ☐ Full Symbolic & Source-Level debug
- ☐ Complete system includes Emulator, 8051 Macro Cross Assembler, RS-232 cable & power supply

MetaLink Corporation  
325 E. Elliot Road  
Chandler, AZ 85225

Phone: (602) 926-0797  
Phone: (800) 638-2423  
Fax: (602) 926-1198

# iceMASTER-RA Development System

MetaLink specializes in enhancing the emulation technology required by EMBEDDED SYSTEMS DESIGNERS. MetaLink has consistently led the industry in emulation technology: The first PC-based 8051 emulator; the first 8052 emulator; the first to offer support for all the unique features of the Dallas High-Speed Microcontroller family components, such as watch-dog-timer, idle modes, power down mode, DMA and A/D.

MetaLink is a full-service emulation company. We support our customers over the long term with services such as repair, discounted upgrades, rental units, 10-day trial purchase periods and free technical support for applications problems. A network of world-wide sales and service representatives is augmented by a well-trained telemarketing staff at headquarters.

## ❑ HOST SPECIFICATIONS

Host IBM PC, XT, AT, 386, 486, PS/2, Laptop, Notebook or compatible system, 640K bytes of RAM, Hard Disk Drive, Monochrome or Color Display, with a Standard RS-232 Serial Port. Operating System DOS 2.0 or greater.

## ❑ EMULATOR SOFTWARE SPECIFICATIONS

File Formats Supported with Symbolic or Source-Level Debugging:

2500AD, Archimedes, Avocet, BSO/Tasking, Franklin, IAR Systems, Intel OMF, Keil, MCC, Microtec Research, Motorola 'S' Record, Systronix and Intel HEX.

Source/Symbol Support:

Assembler, BASIC, C and PL/M Languages

Symbolic or Source-Level Debug:

Setting of Breakpoints  
Setting of Trace ON/OFF  
Viewing of Trace Buffer  
Viewing of Source Window Display  
Viewing of Performance Analyzer  
Assembly/Disassembly of Code

HLL Structure/Content Display of:

Modules Line Numbers  
Scopes Program Variables

## ❑ EMULATOR HARDWARE SPECIFICATIONS

iceMASTER-RA MODELS:

Emulator Base with:

16K Trace Buffer  
Uses Interchangeable Probe Cards  
Communicates at 57K BAUD with PC  
Maximum operating frequency of 33 MHz

Probe Cards:

DS80C320 Probe Card operates up to 25 MHz  
DS87C530 Probe Card operates up to 33 MHz

## ❑ OPERATING CHARACTERISTICS

Clock frequency user-selectable between external target (crystal/clock or internal clock source)  
Real-time, Electrically and Operationally Transparent

## ❑ USER INTERFACE

Keyboard or Mouse Control  
Pull-down & Pop-up menus with fill-in boxes  
Available Main Screen Windows:

Registers and PSW bits  
Bit Memory  
Stack data displayed in HEX &/or ASCII  
Up to 5 Internal Data Memory displayed in HEX &/or ASCII  
Up to 5 External Data Memory displayed in HEX &/or ASCII  
Up to 5 Code Memory displayed in HEX &/or ASCII  
Source Program displayed in Code, HLL Source or Mixed  
Watch window for variable data  
System Status data

Main Screen Window Display Controls:

Movable Selectable (On/Off)  
Sizable Color selection  
Scrollable Highlighting of key data

Function/Hot Key Access:

User-assignable for commands  
User-displayable for quick reference

## ❑ MEMORY OPERATIONS

Emulation Memory:

Program Memory: 64K  
External Data Memory: 64K

Mapping Resolution:

Program: Down to 1-byte  
External Data: Down to 1-byte

Program Memory:

Single Line Assembler (full instruction set support)  
Disassemble in Code or Source/Code mode  
Disassembly may be written to a disk file

Data Memory:

Internal or External Memory  
Fill a block of memory with data  
Copy a block of data to another area  
Change a single address data content  
Compare any two blocks of addressed data  
Displayed data may be written to a file

Registers/SFRs/Bit Memory:

Examine or Modify  
Program Variables:

Examine or Modify

## ❑ EMULATION CONTROLS

Reset from Emulator and Go  
Reset from Target and Go  
Reset Processor  
Go from current Program Counter  
Go From a new Program Counter  
Go Until a Program Counter/Label  
Slow Motion (Repetitive Step commands)  
Step by machine instruction  
Step by Line Number  
Step Over calls  
Step To next function or procedure

## ❑ HARDWARE BREAKPOINTS

Up to 64K real-time Program Addresses  
Up to 64K real-time External Data Addresses

## ❑ TRIGGER CONDITIONS

Set directly in Source window or Pull-down menu  
PC address & range of addresses  
Opcode Value  
Opcode Class  
SFRs/Registers  
Direct byte address & range of addresses  
Direct bit address & range of addresses  
Immediate operand value  
Read/Write to bit address  
Register address modes  
Read/Write to Register address  
Logical AND/OR of any of the above  
External Data address & range of addresses  
Break Count Overflow  
External (CLIP) Break Input  
External (CLIP) Trigger Output

## ❑ TRACE

Real-time trace with view while executing code  
16K x 32-bit Frame Trace Buffer  
Start, Center, End and Variable Trace Trigger settings  
Up to 64K Trace ON/OFF triggers for trace filtering  
Trace Contents consist of:

16-bits Address Bus  
8-bits Data Bus  
8-bits External Clips

Trace Display Modes:

Raw Hex  
Symbolic  
HLL Source  
Mixed

External Clips display format:

Binary data  
HEX data  
Digital waveform

Trace Buffer Operations:

Write trace buffer to a disk file  
Search trace buffer for labels & addresses

## ❑ PERFORMANCE ANALYZER

Program profiling capability

7 year duration

Display options:

Bar Graph  
Frequency Count

Display Modes:

Raw HLL Source Lines  
Symbolic Mixed

Up to 999 Bin capacity

User-controlled Bin set-up:

By Address By Symbol  
By Module By Line Number  
Automatic

## ❑ HELP

On-Line  
Context sensitive  
Hypertext/Hyperlinked

## ❑ DIAGNOSTIC SELF-TEST

Determines Emulation Hardware Status

## ❑ MACRO

Repetitive routines  
User-created and callable

## ❑ ELECTRICAL SPECIFICATIONS

Input Power (maximum):  
1.5 A @ +5 VDC +/-5%  
Power Source: Power supply

## ❑ MECHANICAL SPECIFICATIONS

Emulator Dimensions:

4.4" x 3.25" x 0.9"  
11.18cm x 8.25cm x 2.29cm

Probe Card Dimensions:

4.15" x 3.20" x 1.9"  
10.54cm x 8.13cm x 4.83cm

Emulator weight:

0.75 lbs 0.45 kg

## ❑ ANNUNCIATORS

Emulation LED  
Reset LED  
Reset switch

## ❑ WARRANTY

One (1) year limited warranty, parts and labor

Call Today

1(800) 638-2423

1(800) METAICE or contact your local distributor

Rental plans are available.

Product names are used to purposes of identification only and may be trademarks or registered trademarks of their respective companies.

# Mid-Tech USA

## Embedded Tools

Mid-Tech's single board computers are specially engineered for reliable high-speed operation. Our high-performance computing engines will run your 8031 code faster than any other single board computer on the market. 8031 compatibility means you'll get your designs out the door faster too since you get to keep your existing 8031 code libraries and your favorite development tools, not to mention your hard-won 8031 know-how.

If you prefer a fully-integrated tool set, we offer development software specifically designed to work with Dallas Semiconductor's silicon and Mid-Tech's hardware. As a result of our alliance with Dunfield Development Systems, we offer a variety of software products that deliver a powerful, low-cost development environment. The cross assembler, windowed simulator and compatible debugger ease program development and testing. A low-cost "C" language developer's kit rounds out the inexpensive, full-featured development package.

*Our Developer's Kit includes an update to Dunfield Development Systems' popular 8051 MICRO-C compiler which adds special features for use with the Dallas high-speed processors and Mid-Tech single board computers. New additions include:*

*1) A new library which takes advantage of the DUAL data pointers available in the DS80C320 family. This dramatically speeds up the memory-to-memory operations in those memory models that use external memory.*

- New ASM320 assembler, which supports the additional special function registers of the DS80C320 family.*
- New CC320 compile command which uses the DS80C320 code library.*
- New 80320.IDE file for the integrated environment.*
- New 80320REG.H file which defines ALL DS80C320 SFRs to the compiler.*

*2) Additional libraries tailored specifically for each Mid-Tech Single Board Computer. This allows you access to each system's peripheral set in a consistent and straightforward manner regardless of the particular implementation details.*

- Analog and digital I/O*
- Real-time clock and timers*
- Nonvolatile memory*
- I<sup>2</sup>C and Microwire peripherals*
- LCD, keypad, indicators*

---

**Mid-Tech** Computing Devices USA P.O. Box 218 Stafford, CT 06075 Tel: (203) 684-2442

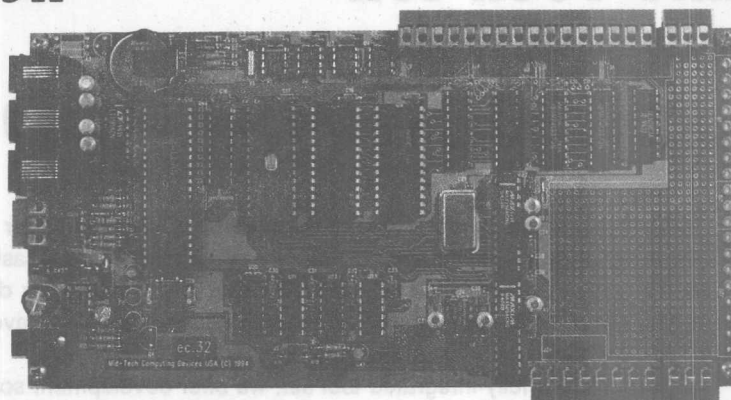
---



# Mid-Tech USA

## ec.32

### Fast Multipurpose Embedded Computer



Mid-Tech's ec.32 multipurpose single board computer makes an ideal low-cost development system, a full-speed DS80C320 evaluation vehicle, or a general-purpose embedded computer. The Dunfield software get you on the air fast with an easy-to-use simulator, debugger, and assembler. Our "C" developer's package provides everything you need to work with the DS80C320 and offers full library support for all on-board and remote peripherals. If you prefer a different environment, integrate the included "C" and assembler source drivers directly into your application programs.

#### Processor

High-performance DS80C320  
Crystal frequency 25 MHz  
Throughput 6 MIPS  
Internal RAM 256 bytes

#### Memory

Low-power CMOS EPROM/RAM  
Program PROM 32K  
Program RAM 32K  
Data RAM 32K  
RAM backup .47F supercap  
(lithium and NiCd options available)

#### Parallel I/O

Sixteen TTL inputs  
Input levels LS TTL

#### Eight TTL outputs

Output levels LS TTL

#### Eight Darlington outputs

Sink capability .5A @ 50V

#### Fourteen pseudo bidirectional

Input levels quasi TTL  
Output levels quasi TTL

#### Indicators

Power LED  
Program load LED  
General-purpose indicator LED

#### Analog I/O

4-channel ADC  
Simultaneous track-and-hold  
Single-ended or differential  
Resolution 8 bits  
Input voltage 0-2.50V  
Conversion time 3.6  $\mu$ S

#### 4-channel DAC

Simultaneous update capability  
Resolution 8 bits  
Output voltage 0-2.50V  
Settling time 6  $\mu$ S

#### Serial I/O

Two Independent High-speed UARTs  
Port 0 RS232 or RS485  
Port 1 RS232

#### Timers/Counters

Three Independent 16-bit timer/counters  
Resolution 480 nS or 160 nS  
(individually selectable)

#### I<sup>2</sup>C subsystem

Real-time clock/calendar/timer with RAM  
RTC resolution 1 mS  
Programmable interval timer  
Nonvolatile RAM 256 bytes

#### Nonvolatile storage

EPROM 512 bytes

#### External I<sup>2</sup>C port

Connector RJ11 jack

#### Power

Switch-mode regulator  
Input voltage 8-35 VDC  
Power dissipation 2 watts  
Connector 2.5mm barrel

#### Expansion

Pad-per-hole prototype area  
I<sup>2</sup>C port for external peripherals  
Access to all on-board I/O points

#### Environmental

Dimensions 4.5" x 9"  
Operating temp. 0-70°C  
Storage temp. -40-85°C  
Humidity 0-95% (non-condensing)

**Mid-Tech** Computing Devices USA P.O. Box 218 Stafford, CT 06075 Tel: (203) 684-2442

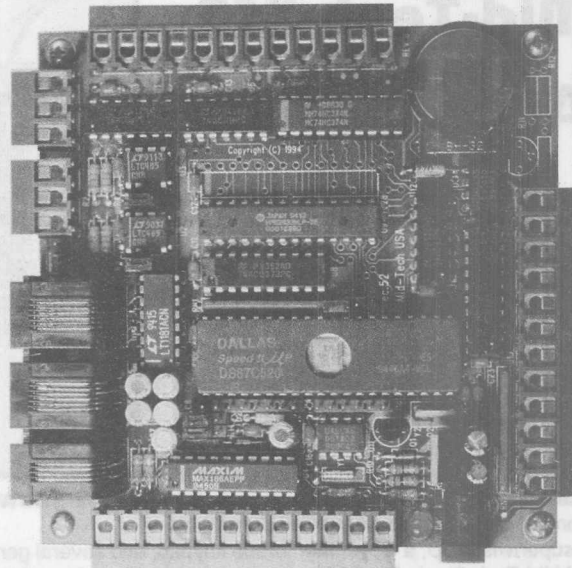


# Mid-Tech USA

## ec.52

### Very High-speed Embedded Computer

The fully CMOS ec.52 will run your 8031 code faster than any other single board computer on the market. You will get even greater performance gains using the ec.52 in conjunction with the optimized Dunfield "C" development tools. And to get your code up and running quickly, we offer a PC-hosted simulator/debugger package that includes a DS87C520 assembler. "C" and assembler source drivers for all local and remote peripherals included.



#### Processor

High-performance CMOS DS87C520  
Crystal frequency 33MHz  
Throughput 8 MIPS  
(29.9152 MHz option available)  
Internal RAM 256 bytes

#### Serial I/O

Two independent high-speed UARTs  
Port 0 RS232 or RS485  
Port 1 RS232 or RS485

#### Parallel I/O

Eight CMOS (AC) inputs  
Input levels AC MOS

Eight CMOS (HC) outputs  
Output levels HCMOS

Fourteen pseudo bidirectional  
Input levels quasi TTL  
Output levels quasi TTL

#### Memory

High-speed program/data storage  
Program PROM 16K  
Data RAM 1K  
External RAM 32K  
(usable as program and data memory)  
RAM backup .47F supercap  
(lithium and NiCd options available)

#### Analog I/O

8-channel ADC  
Single-ended or differential  
Resolution 12 bits  
Input voltage 0-4.096 V  
Conversion time 6  $\mu$ S/channel  
Acquisition time 1.5  $\mu$ S  
Sample rate 133 KHz  
DC impedance 12 K ohm

#### Timekeeping

Real-time clock/calendar with RAM  
RTC resolution 1 second  
RTC RAM 24 bytes

#### Timers/Counters

Three independent 16-bit timer/counters  
Resolution 363 nS or 121 nS  
(individually selectable)

#### Indicators

Power low current LED

#### Expansion

PC port for external peripherals  
Access to all on-board I/O points

#### Power

Low dropout pass stage  
Input voltage 5.2-16 VDC  
Current 150mA

#### Environmental

Dimensions 4.25" x 4.25"  
Operating temp. 0-70°C  
Storage temp. -40-85°C  
Humidity 0-95% (non-condensing)

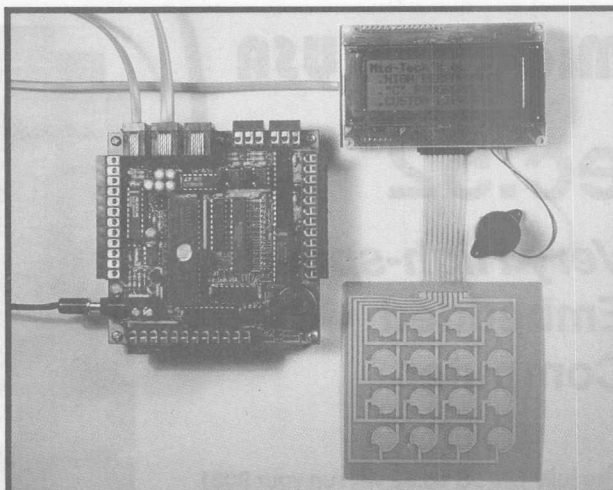
**Mid-Tech** Computing Devices USA P.O. Box 218 Stafford, CT 06075 Tel: (203) 684-2442

## Mid-Tech USA Accessories

Mid-Tech offers a number of accessories that afford our single board computers true plug-and-play capability. The lineup includes communication cables, power supplies, networked I/O, and a variety of digital and analog peripheral devices. These are in addition to our full line of PC-hosted software products.

Available from Mid-Tech is an addressable I<sup>2</sup>C peripheral that connects to any standard I<sup>2</sup>C port where a local or remote user I/O interface panel is required. A number of useful functions are supported using just two general purpose processor I/O lines: a 20 by 4 supertwist LCD, a 4 by 4 membrane keypad, and several general purpose digital I/Os (for LEDs, beepers, etc.). All Mid-Tech Single Board Computers support this universal peripheral.

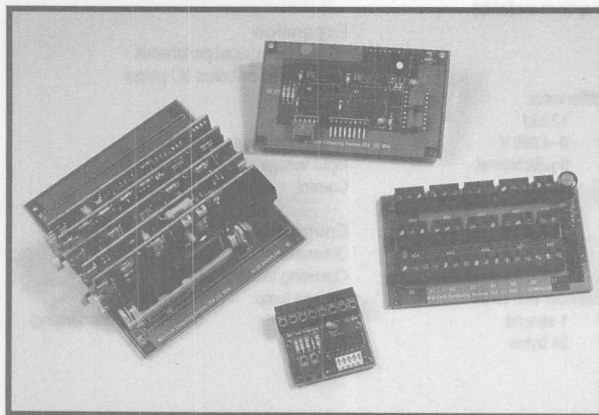
A general purpose, firmware-based, I<sup>2</sup>C driver coded in generic 8051 assembler is included along with various second level function modules written in both assembler and "C". Replacement "C" functions for putchar and getkey along with other useful support routines are included to let you transparently access this peripheral from your application program.



ec.52 shown with optional I<sup>2</sup>C user I/O device

## Other Products

Mid-Tech has been extensively involved with battery-backed and battery-operated systems. Our DS2250-based ec.25 is a complex data collection instrument with power management capability that allows it to operate from battery power for periods of months, even years. Supporting a very complete analog and digital peripheral set, the ec.25 is essentially an open frame data logger suitable for use as the basis for a variety of data capture and control oriented OEM products.



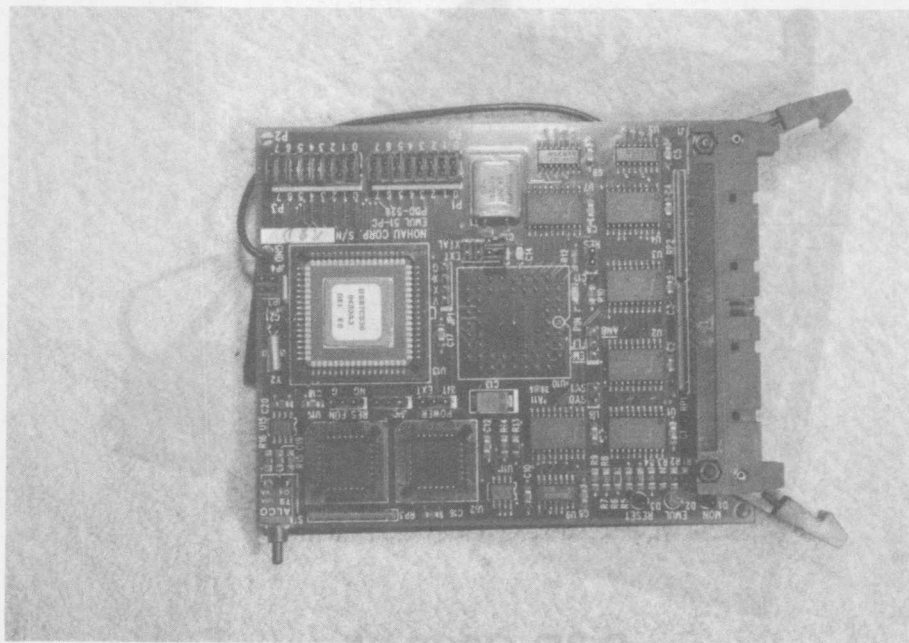
ec.25 data collection computer

## Coming Attractions

Mid-Tech is committed to providing high-quality, low-cost support products for Dallas Semiconductor's high-speed processor family. Presently under development is the *ec.32 lite*. This is a high-density, miniaturized version of our popular ec.32 embedded computer. In response to customers' requests, we are also transforming some of our general purpose instruments to serve more specialized functions. Look for a very high-performance communications controller based on our high-speed ec.52 core. Mid-Tech will, of course, support all new Dallas processors as they become available.

**Mid-Tech** Computing Devices USA P.O. Box 218 Stafford, CT 06075 Tel: (203) 684-2442

# NOHAU CORPORATION



## POD-C520-33

The POD-C520-33 supports the **DS87C520** and **DS87C530**. Use it together with the Nohau emulator board, the EMUL51-PC/EA256-C520-BSW-33. The emulator board is a PC plug-in board and connects to the POD-C520 with a five-foot (1.5m) cable. Jumpers on the POD allow the selection of the microcontroller's crystal or clock, power, operational speed, real-time clock oscillator input, and traced pins (with optional trace board).

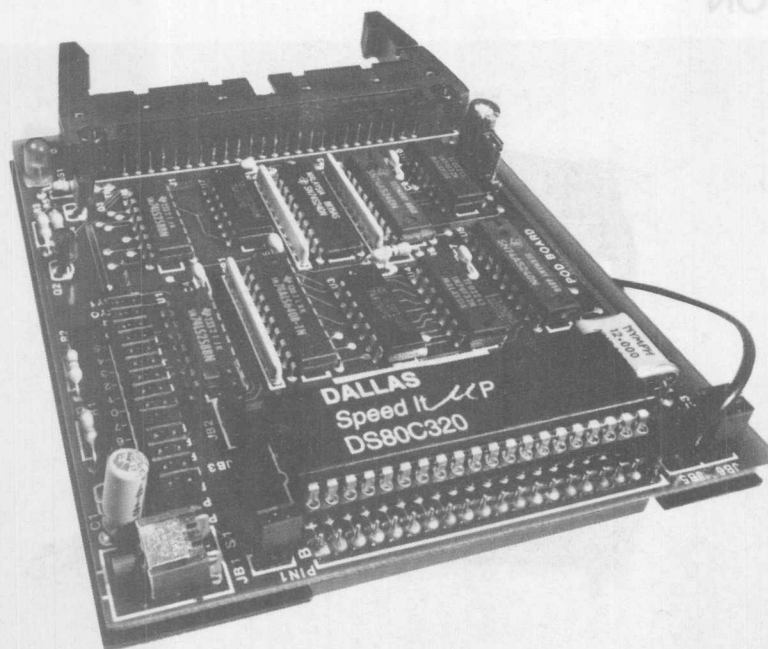
This pod terminates to a 52-pin Pin-Grid-Array (PGA). Nohau carries various adapters to support the DS87C520 in both the DIP and PLCC packages. There is also an adapter for the DS87C530 in the PLCC package. These adapters are the EDI/PG52/DP40-C520, EDI/PG52/PL44-C520, and EMUL51-PC/PGA52-PLCC52.

The pod has an on-board crystal that allows the user to operate this pod "stand-alone," without being connected to a user's target board.

<b>EDI/PG52/DP40-C520</b>	52-pin PGA to 40-pin DIP adapter for DS87C520.
<b>EDI/PG52/PL44-C520</b>	52-pin PGA to 44-pin male LCC adapter for DS87C520.
<b>EMUL51-PC/PGA52-PLCC52</b>	52-pin PGA to 52-pin male LCC adapter for DS87C530.

Nohau Corp. 51 E. Campbell Ave Campbell, CA 95008 (408) 866-1820 FAX: (408) 378-7869

040595 12/15



## POD-C320-25

The POD-C320-25 supports the **DS80C320**. Use it together with the Nohau emulator board, the EMUL51-PC/EA256-C320-BSW-25. The emulator board is a PC plug-in board and connects to the POD-C320 with a five-foot (1.5m) cable. Jumpers on the POD allow the selection of the microcontroller's crystal or clock, power, operational speed, and traced pins (with optional trace board).

This pod terminates to a 40-pin Dual-in-Line Package pinout. Nohau carries an adapter to convert this pinout to a 44-pin PLCC male plug, allowing the user to plug into a chip carrier socket.

This pod has an on-board crystal that allows the user to operate this pod "stand-alone," without being connected to a user's target board.

## EMUL51-PC/DIP40-PLCC44

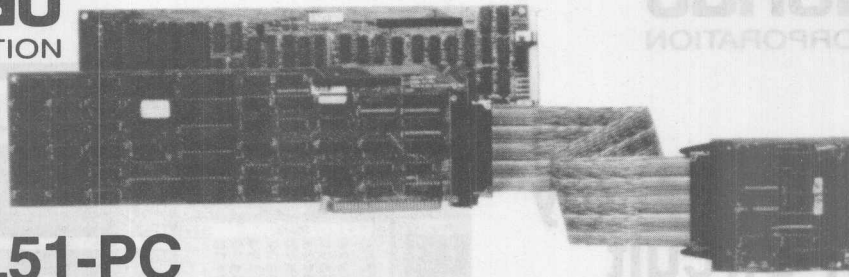
Adapter for 44-pin PLCC. This adapter connects to the bottom of a pod with a 40-pin DIP male set of pins. The bottom is a precision-milled block with gold pins that allows connection to a 44-pin PLCC socket.

---

Nohau Corp. 51 E. Campbell Ave Campbell, CA 95008 (408) 866-1820 FAX: (408) 378-7869



**NOHAU**  
CORPORATION



## EMUL51-PC

### Saves time on all 8051 family projects.

EMUL51-PC is an in-circuit emulator dedicated to the 8051 family. The EMUL51-PC truly emulates the microcontrollers from all manufacturers of 8051 derivatives. This means that your EMUL51-PC will work just like your microcontroller when it is plugged into your target.

#### System Specification

##### Host

IBM PC/XT/AT, PS/2 or compatible.  
Minimum 640K of RAM.  
SUN, HP and other workstations

##### External Serial Box

The emulator boards can be installed in an external box with serial communication to PC. Up to 115K baud supported.

##### Languages Supported

Third party assemblers, PL/M-51 and C-51 compilers.

##### High Level Debugging

Window for source level debugging. Single Step or Line Step with breakpoints marked directly in the code. Full support for local or global variables in C-51.

##### In-line Assembler and disassembler

Full instruction set and symbols supported.

##### Symbolic Support

Full symbolic debugging and type checking. Same symbols can be used in different modules.

##### File formats Supported

Intel HEX/OBJ/OMF/SYM, Avocet, Archimedes/IAR Franklin/Keil, BSO/Tasking, Intermetrics/Whitesmiths.

##### Real time Emulation

Full speed emulation up to 42 MHz. No wait states and no intrusion on memory, stack, I/O or Interrupt pins.

##### Emulation Memory

64K XDATA memory and 64K CODE memory. Up to 256K with bankswitch option.

##### Memory Mapping

Mappable in 4K pages.

##### Macro

Test session automation and macro command definition. IF/ELSE, REPEAT/WHILE structures.

##### Debug Session Logging

Record emulation session and all setups to a file.

##### Breakpoints

64 K program breakpoints.  
64 K data read and write breakpoints.  
Break on external signal.  
Break on direct access to internal bit or byte memory.  
Break on a range of addresses.

With the trace board option you can break on any 48 bit combination of address, data, RD, WR, OP code fetch, interrupt level, ports of external signals.

##### Single Stepping

Single or multiple instruction stepping. Step over calls and interrupts. Line stepping in high level languages.

##### Execution timer

Resolution down to half a cycle.

##### Real Time Trace (optional)

256K deep by 64 bits wide with time stamp.

##### Trace operation

With each bus cycle, 48 bits of address, data, ports and external signals are compared with eight independent "48 bit registers" to produce eight "condition signals". These eight conditions are then combined with six bits from a "state-machine". The combination of these 14 signals creates a new state (or retains the old state). It also produces a TRIG and a FILTER signal that are used to control what is saved in the trace buffer. If the eight condition signals are named A, B, C, D, E, F, G and H an example of how this could be used follows:

A THEN B THEN C THEN D THEN E THEN F THEN G THEN H where A - H could be code addresses.

The trace buffer can be viewed, reprogrammed and restarted without affecting emulation.

##### Trigger Breakpoint

The 48 bit trace trigger qualifiers can be used to define complex breakpoints also on program execution.

##### Trace Display

Display trace in disassembled symbolic or binary/hex form, or as high level source code. Display can be saved to a file. Trace can be started, stopped and displayed independent of program execution.

##### Program Performance Analyzer

Histogram and statistical information of program execution in real time.

### Devices Supported:

DS5000,

DS5000T,

DS5000FP,

DS5001FP,

DS5002FP,

DS80C320,

DS87C520,

DS87C530

Nohau Corp. 51 E. Campbell Avenue Campbell, CA 95008 (408) 866-1820 FAX: (408) 378-7869



**NOHAU**  
CORPORATION

## 8051 Family In-Circuit Emulator

The EMUL51-PC is a high performance in-circuit emulator specifically designed to give an optimized environment to develop your 8051 family microcontroller hardware and software.

The EMUL51-PC consists of a board which plugs directly into the IBM PC/XT/AT bus. The optional Trace board features an advanced trace function with sophisticated trigger capabilities.

The POD, which plugs into the target system, is connected with a 5 ft. ribbon cable to the emulator board to provide a flexible operating range.

Optionally an RS-232 box can be used. It communicates with the PC at up to 115K baud. Yet another option, the LanICE, makes EMUL51-PC run on workstations like SUN or HP.

### The World's Most Popular 8051 Emulator.

Since its introduction in 1986 Nohau has delivered over 10,000 EMUL51-PC emulators. And each emulator is often used in several projects where different 8051 derivatives are used, only a change of the probe is required when a new derivative needs emulation support.

### Choice of Different User Interfaces.

Early in the evolution of the EMUL51-PC's user interface it became clear that each customer had different opinions of how they would like the interface to work and what features were important to them. On these pages we show one of the three main user interface choices for the EMUL51-PC: Microsoft Windows 3.x. ChipView's Borland Keypress compatible and Nohau's original pull-down/command line are not shown. Recognizing EMUL51-PC's enormous popularity, several third

party compiler vendors have ported their user interfaces to the EMUL51-PC to provide emulator interfaces similar to their simulator interfaces. Among these vendors are: Intermetrics/ Whitesmiths, Keil/Franklin and Production Languages Corporation.

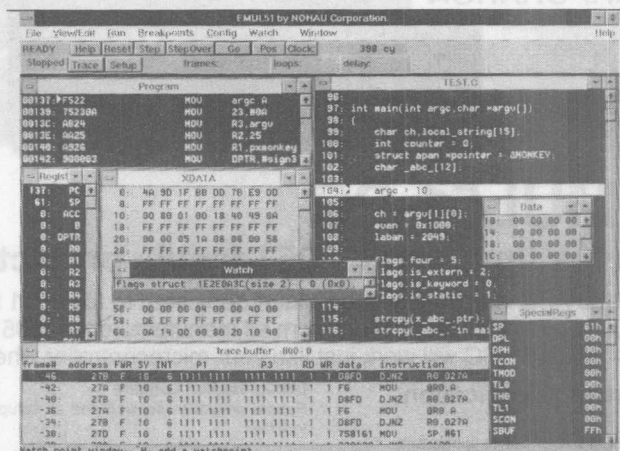
### Hosted on PCs and Workstations

The emulator was designed to be plugged into a full size PC AT style slot. The optional trace needs a second slot. These same boards can also be supplied in an "RS-232 box" which communicates with the PC over a standard COM port. To use the EMUL51-PC on XWindows workstations such as SUN or HP, the Nohau LanICE is available. Because LanICE uses a high speed (10 Mbit/second) local area network, not only can it be placed far away from the workstation, but it maintains the relatively high code loading speed of the Nohau emulators plugged into your PC on your desk top.

LanICE also supports personal computers on a network. This makes LanICE useful where several people must share the emulator (and one target system) but have different offices. Several engineers can time-share the prototype target board and emulator without moving any equipment.

### Real time Trace

The EMUL51-PC offers trace features not found in other emulators. For instance the trace buffer can record up to 256K bus cycles with 64 bits of data. The trace can be operated "on-the-fly" which means that it can be viewed, programmed and retriggered without disturbing program execution. With the trace set-up menu you can define exactly what events are to be stored in the trace buffer. The real-time trace can be stopped (triggered) at a selected event or after a combination of multiple events. By aligning the trigger point anywhere in the trace buffer you have a full choice of pre- and post triggering.



Nohau Corp. 51 E. Campbell Avenue Campbell, CA 95008 (408) 866-1820 FAX: (408) 378-7869

# INDEX

- address recognition, 34
- addressing mode, register indirect, 31
- addressing modes, 54
  - direct, 55
  - extended, 55
  - immediate, 55
  - page, 55
  - register, 55
  - register indirect, 55
  - register indirect with displacement, 55
  - relative, 55
- ALE noise generation, 380
- application note
  - 55: The D88C320 as a Drop-In Replacement for the 80C2, 384
  - 57: D88C320 Memory Interface Timing, 388
  - 75: Using the High-Speed Micro's Serial Port, 393
  - 78: Using Power Management with the D88C320, 395
  - 79: Using the D88C320 Real Time Clock, 397
  - 80: Using the High-Speed Micro's Watchdog Timer, 398
  - 81: Memory Expansion with the High-Speed Micro-controller Family, 381
  - 82: High-Speed Micro Memory Interface Timing, 374
  - 91: Microcontroller Design Guidelines for Reducing ALE Signal Noise, 380
- assembler, 54, 355
- band-gap reference, 5, 28, 45, 75, 80, 85, 93, 125
- bank switching, 7, 381
- battery, 138
- beaded film, 93, 138, 140
- life, 140
- selection, 139
- bezel case, 107
- formulas, 128, 129
- generator, Timer 2, 113
- maximum, 305
- Mode 0, 119
- Mode 1, 119
- Mode 2, 119
- Mode 3, 119
- PWM effect on, 57
- bit addressable locations, 7
- broadcast address, 131
- brownout, 83, 95
- buffered board system (BBS), 355
- burst mode operation, 324
- C language, 355
- calendar, 93
- cache loading, 55
- clock source, 5, 27, 55, 84, 93, 91
- configurable, 54, 395
- configuring port pins as inputs, 105
- counter (Timer/Counter), 104, 110, 113
- crystal
- CPU, 55
- RTC, 138, 139, 345, 353
- troubleshooting, 148
- warm-up period, 39, 85
- D
  - data logging example, 324, 327
  - data memory, 7, 35, 55
  - interface, 70, 75
  - internal, 68
  - software access, 73
  - status cycles, 75, 76
  - timing, 75
  - data printer (DPT), 4, 55, 73, 141, 144, 146
  - data retention, 64, 138, 139
  - data sheets
    - D88C310, 125
    - D88C320, 174
    - D88C321, 281
    - D88C322, 295
    - D88C323, 308
    - D88C324, 345
  - development tool, 385
  - direct addressing, 55
  - drop-in compatibility, 180
- E
  - electrostatic discharge (ESD), 133
  - EPROM
    - "turn-off" time, 591
    - interface, 70
    - programming, 68
    - speed selection, 398
    - System Control Byte, 38
    - expanded memory, 68, 100, 105
    - extended addressing, 55
    - external interrupt, 5-interrupt, external
    - external reset, 55a reset, external
- F
  - feature summary, 5
  - firmware security, 39

# A

access time, 75, 100  
address recognition, 34  
addressing mode, register indirect, 31  
addressing modes, 54  
    direct, 55  
    extended, 56  
    immediate, 56  
    page, 56  
    register, 55  
    register indirect, 55  
    register indirect with displacement, 56  
    relative, 56  
ALE noise generation, 380  
application note  
    56: The DS80C320 as a Drop-In Replacement for the 8032, 284  
    57: DS80C320 Memory Interface Timing, 286  
    75: Using the High-Speed Micro's Serial Ports, 293  
    78: Using Power Management with the DS87C5x0, 306  
    79: Using the DS87C530 Real Time Clock, 333  
    80: Using the High-Speed Micro's Watchdog Timer, 355  
    81: Memory Expansion with the High-Speed Microcontroller Family, 361  
    89: High-Speed Micro Memory Interface Timing, 374  
    91: Microcontroller Design Guidelines for Reducing ALE Signal Noise, 380  
assemblers, 54, 386

# B

band-gap reference, 5, 28, 45, 79, 80, 82, 83, 132  
bank switching, 7, 361  
battery, 139  
    backed bits, 93, 136, 140  
    life, 140  
    selection, 139  
baud rate, 107  
    formulas, 123, 124  
    generator, Timer 2, 113  
    maximum, 305  
    Mode 0, 119  
    Mode 1, 119  
    Mode 2, 119  
    Mode 3, 119  
    PMM effect on, 87  
bit addressable locations, 7  
broadcast address, 131  
brownout, 83, 92  
bulletin board system (BBS), 386  
burst mode operation, 324

# C

C language, 386  
calendar, 53  
capacitive loading, 58  
clock source, 5, 27, 59, 84, 89, 91  
compiler, 54, 386  
configuring port pins as inputs, 102  
counter (Timer/Counter), 104, 110, 113  
crystal  
    CPU, 58  
    RTC, 136, 138, 349, 353  
    troubleshooting, 149  
    warm-up period, 39, 85

# D

data logging example, 324, 337  
data memory, 7, 38, 55  
    interface, 70, 72  
    internal, 68  
    software access, 73  
    stretch cycles, 75, 76  
    timing, 75  
Data Pointer (DPTR), 4, 56, 73, 141, 144, 146  
data retention, 93, 138, 139  
data sheets  
    DS80C310, 152  
    DS80C320, 174  
    DS80C323, 281  
    DS83C520, 282  
    DS87C520, 206  
    DS87C530, 243  
development tools, 386  
direct addressing, 55  
drop-in compatibility, 150

# E

electrostatic discharge (ESD), 133  
EPROM  
    "turn-off" time, 291  
    interface, 70  
    programming, 68  
    speed selection, 288  
    System Control Byte, 38  
expanded memory, 68, 100, 102  
extended addressing, 56  
external interrupt. *See* interrupt, external  
external reset. *See* reset, external

# F

feature summary, 6  
firmware security, 39

framing error, 21, 29, 35, 129  
 fundamental mode crystal, 58, 149

**G**

given address, 131

**I**

I/O, 4, 19, 26, 31, 33, 68, 69, 100, 101  
   configuring as input, 102  
   current-limited transitions, 102  
   maximum drive current, 101  
   optional functions, 103  
   output functions, 101  
   port timing, 102  
   ports during Idle mode, 83  
   ports during reset, 93  
   ports during Stop mode, 83  
 Idle mode, 21, 82  
   exiting, 82  
   using watchdog timer with, 81  
 immediate addressing, 56  
 In-System Disable (ISD), 94  
 instruction set, 54, 59, 141  
 instruction timing, 54, 59, 60, 61, 64, 102  
 instruction timing comparison, 64  
 Interrupt, external, 82, 91, 96, 98  
 interrupt, 95  
   acknowledge, 98  
   changes in Stop mode, 91  
   exiting switchback, 88  
   global enable/disable, 96  
   latency, 98  
   power fail, 79, 97  
   priority, 95, 97  
   real time clock, 97  
   sensitivity in Stop mode, 91  
   serial, 97  
   service routine, 95  
   simulated, 97  
   sources, 96  
   timer, 96  
   vector, 95  
   vector table, 95  
   watchdog, 81, 97, 116

**L**

lithium battery, 139

**M**

memory  
   expansion beyond 64KB, 7, 361  
   interconnect, 70  
   interface timing, 286, 374  
   internal data, 68

internal program, 68  
   map, 7, 8  
   organization, 7  
 MOVX timing, 60, 75, 76, 77, 78  
 multiplexed bus, 5, 19, 70, 100  
 multiprocessor communication, 131  
   concerns with PMM, 89, 314  
   using address recognition, 131  
   using the SM2 flag, 29, 35

**N**

no-battery, 93  
 nonvolatile SRAM, 7, 68, 69, 139

**O**

ordering information, 3  
 oscillator  
   crystal, 58, 315  
   external, 58, 315  
   ring, 84, 89, 91, 315  
   startup delay, 317

**P**

page addressing, 56  
 parallel I/O, 100  
 parity  
   bit in serial communication, 119, 126  
   flag, 44, 57  
 peripherals  
   memory mapped, 70  
   serial interface, 119  
   slow access, 75  
 Port 0  
   description, 100  
   SFR, 19  
 Port 1  
   description, 101  
   SFR, 26  
 Port 2  
   description, 100  
   SFR, 31  
   use in Register Indirect addressing, 55, 73  
 Port 3  
   description, 101  
   SFR, 33  
 power consumption, 2, 82, 84, 85, 91, 136, 285, 288  
 power fail  
   interrupt, 79  
   monitoring in Stop mode, 80  
   reset, 79  
 power management, 306–333  
   features, 79  
   mode timing, 87  
   modes, 85

## R

### RAM

- direct, 7, 55
- indirect, 7, 55
- internal MOVX, 68
- MOVX, 7
- scratchpad, 5, 7

### real-time clock (RTC)

- accuracy, 138, 149
- alarm, 137, 335
- calibration, 138, 349
- crystal, 138
- day of the week, 138
- disabling, 136
- enabling, 136
- interface software example, 341
- interrupt, 97
- reading, 136, 335
- setting, 136, 335
- software tricks, 337

### register. See special function register

### register addressing, 55

### register indirect addressing, 55

### register indirect with displacement addressing, 56

### register map, 7, 8

### relative addressing, 56

### reset, 27

- clock source after, 89
- determining source, 149
- external, 83, 92
- no-battery, 140
- power fail, 92
- power on, 27
- power-on, 68, 79, 92
- ROMSIZE setting after, 69
- sensitivity in PMM, 310
- state, 93
- Stretch setting after, 75
- vector, 68
- watchdog, 92, 104, 115, 116

### ring oscillator

- "switchback", 319
- operation following Stop, 84, 91
- stability, 89, 91
- sustained operation, 91

### ROMSIZE feature, 69, 371

### RS-232. See serial port

### serial port

#### address recognition, 34, 89, 296

#### and PMM, 87

#### asynchronous mode, 119

#### baud rate. See baud rate

#### dual serial port example, 295

#### framing error, 129

#### initialization, 120

#### mode description, 124, 126, 129

#### multiple devices. See multiprocessor communication

#### parity bit, 119

#### polled vs. interrupt, 294

#### switchback, 88

#### synchronous mode, 119

#### troubleshooting, 149

### software compatibility, 386

### software timing loops, 87, 150, 284

### special function register

#### A / ACC, 46

#### B, 47

#### CKCON, 25

#### DPH, 20

#### DPH1, 20

#### DPL, 19

#### DPL1, 20

#### DPS, 20

#### EIE, 46

#### EIP, 48

#### EXIF, 27

#### IE, 31

#### IP, 33

#### P0, 19

#### P1, 26

#### P2, 31

#### P3, 33

#### PCON, 21

#### PMR, 37

#### PSW, 43

#### RCAP2H, 43

#### RCAP2L, 42

#### ROMSIZE, 36

#### RTAH, 48

#### RTAM, 48

#### RTAS, 47

#### RTASS, 47

#### RTCC, 49

#### RTCD0, 53

#### RTCD1, 53

#### RTCH, 52



RTCM, 52  
 RTCS, 51  
 RTCSS, 51  
 SADDR0, 32  
 SADDR1, 32  
 SADEN0, 34  
 SADEN1, 34  
 SBUF0, 30  
 SBUF1, 36  
 SCON0, 29  
 SCON1, 35  
 SP, 19  
 STATUS, 39  
 T2CON, 40  
 T2MOD, 42  
 TA, 40  
 TCON, 22  
 TH0, 24  
 TH1, 24  
 TH2, 43  
 TL1, 24  
 TL2, 43  
 TLO, 23, 24  
 TRIM, 28  
 WDCON, 44  
 special function register map  
   DS80C310, 10  
   DS80C320, 12  
   DS80C323, 12  
   DS83C520, 14  
   DS87C520, 14  
   DS87C530, 16  
 stack, 5, 8, 386  
 Stop mode, 82  
   enabling band-gap reference, 80  
   resuming with crystal oscillator, 83  
   resuming with ring oscillator, 84, 91  
 stretch cycle, 75  
 switchback  
   considerations when using, 312

enabling/initiating, 311  
 interrupt-driven, 88  
 serial port-initiated, 88

## T

technical support, 386  
 timed access  
   examples, 133  
   protected bits, 132  
   protection scheme, 132  
   using, 132  
 timer  
   16-bit, 104  
   auto-reload. *See* timer, Mode 2  
   default time-base, 114  
   interrupt. *See* interrupt, timer  
   Mode 0, 106  
   Mode 1, 106  
   Mode 2, 107  
   Mode 3, 108  
   output clock generator, 114  
   PMM effect on, 87  
   time-base selection, 114  
   timer 2, 108  
   timer 2 modes, 110

## U

UART. *See* serial port

## W

watchdog timer, 115  
   as long interval timer, 358  
   as system monitor, 116, 356  
   block diagram, 115  
   interrupt. *See* interrupt, watchdog  
   period, 116  
   reset. *See* reset, watchdog  
   wake up from Idle mode, 81

